P·R·O·C·E·E·D·I·N·G·S

Speech Recognition Workshop

February 1986

DTIC
SELECTED
APR 0 1 1986
D

s ɛ lə bre t

s ɛ lə bre t

Sponsored by:

DARPA
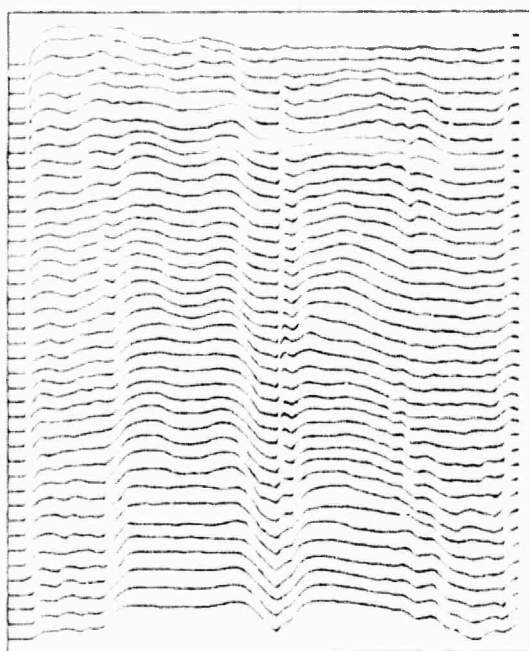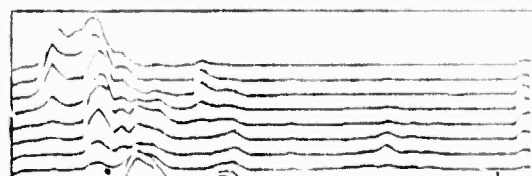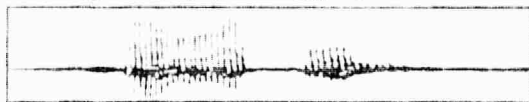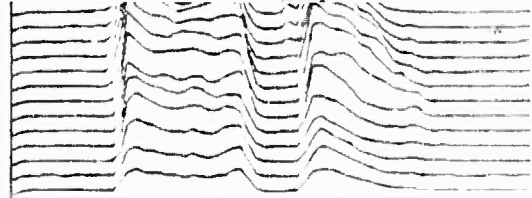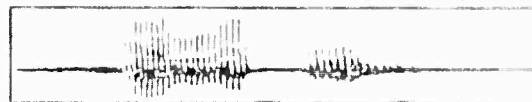
86 3 31 027

Defense Advanced Research Projects Agency
Information Processing Techniques Office

SPEECH RECOGNITION

Proceedings of a Workshop
Held at
Palo Alto, California
February 19-20, 1986

Sponsored by the
Defense Advanced Research Projects Agency

APPROVED FOR PUBLIC RELEASE

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

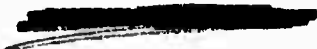| REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SAIC-86/1546	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* SPEECH RECOGNITION Proceedings of a Workshop, February 1986		5. TYPE OF REPORT & PERIOD COVERED ANNUAL TECHNICAL September 1985-February 1986
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) LEE S. BAUMANN (Ed.)		8. CONTRACT OR GRANT NUMBER(s) MDA903-34-C-0160
9. PERFORMING ORGANIZATION NAME AND ADDRESS SCIENCE APPLICATIONS INTERNATIONAL CORPORATION 1710 Goodridge Drive, 10th Floor McLean, Virginia 22102		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA ORDER No. 3456
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE February 1986
		13. NUMBER OF PAGES 120 pps.
14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Speech Recognition; Signal Processing; Acoustic Phonetics; Intelligent Systems; Cochlear Models; Speaker-independent vowel recognition; Pitch tracking; Vowel Coarticulation; Phonological studies; Continuous word recognition; Robust HMM-based techniques for speech recognition.

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This document contains the technical papers for the Speech Recognition Program which were reviewed by the key research specialists from the research activities participating in the program sponsored by the Information Processing Techniques Office, Defense Advanced Research Projects Agency. The reviews of these papers were presented at a workshop conducted on 19-20 February 1986, in Palo Alto, California.

Keywords:

DD ₁ FORM JAN 73 **1473** EDITION OF 1 NOV 65 IS OBSOLETE

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

FOREWORD

A workshop for research personnel involved in the DARPA program on Speech Recognition was held on 19-20 February 1986 in Palo Alto, California. The purpose of the workshop was to review progress on the technical aspects of the work being done by researchers at Carnegie-Mellon University, MIT, BBN, TI, MIT-Lincoln Laboratory, NBS, SRI International, Schlumberger Palo Alto Research, and the Air Force Aerospace Medical Research Laboratory. Also in attendance were personnel from Dragon Systems who have recently been added to the research team; SPAWAR - the contractual monitoring organization; and NOSC, RADC, CIA and NSA - government laboratories and organizations interested in the research.

Commander Allen Sears, the program manager for the DARPA Speech Recognition Program, welcomed the more than forty-five attendees to the workshop. He thanked Gary Kopec of Schlumberger for conducting a hands on working session for interested researchers on the Signal Representation Language (SRL), and Integrated Signal Processing System (ISP) on the previous day, 18 February 1986. SRL and ISP, he noted, are fundamentally programming environments for LISP-based signal processing and may be key tools for researchers in the DARPA program.

Cdr. Sears noted that the essence of the program was cooperation and that all groups were expected to contribute to the successful end results and were not to duplicate research techniques but to use successfully developed items in their systems regardless of who did the original work. Also, he indicated, we must get the service laboratories, like AFWAL and NOSC, involved in demonstrations of the maturing technology. Cdr. Sears reviewed the results of his recent visit to Carnegie-Mellon University to view a demonstration of their speech recognition system. The progress being made was impressive; the next demonstration is scheduled for September 1986 where the goal is to show an order of magnitude improvement in processing performance. In closing, Sears enjoined the group to keep the end objective in mind - a relevant demonstration of results and to observe the essence in this research program -- cooperation.

This proceeding consists of seventeen technical reports which were reviewed by the key individual for that program at the workshop. Also appended is a companion paper by Dragon Systems concerning their work in speech training. The papers are arranged in the order the subjects were

presented using three general groupings: review of new generation system (NGS) for continuous speech recognition, review of NGS for robust speech recognition, and review of supporting infrastructures.

In addition to the government and research attendees, four guest experts were invited to visit the sessions and to present their reactions to the material discussed during the workshop. The invited guests were:

Steven F. Boll   - ITT-DCD
Jordan Cohen   - IDA
Bruce T. Lowerre - Hewlett-Packard Labs
Fred Jelinek   - IBM - TJ Watson Research Center

All four reviewers gave their impressions of the research program in a give and take session on the afternoon of day two of the workshop. The reviewers indicated they were impressed by the progress made to date. In general, they expressed satisfaction with the aims and objectives of the program, with the cooperative nature of the research, and with the tight, but realistic, schedule as outlined in the program plan. Also, it was stated that good and sufficient test data would be required in order to insure the robustness of the resultant systems. The group advised all researchers to include in their plans sufficient flexibility as they are sure to find that the environment at the end of the project will be vastly different from that envisaged at this point in the evolution of the projects. Also, it was stressed that sufficient training time must be built into any schedule for installation and use of a speech system. For, it was stated, a speaker can always make the system fail - sometimes even when not trying to do so. Finally, Dr. Jelinek of IBM warned that the not-invented-here syndrome is hard to break and the research teams need to accept the selected techniques and proceed with new research regardless of which group developed the algorithm or the particular processing code. Cdr. Sears expressed the appreciation of the Defense Advanced Research Projects Agency for the candid and informed remarks of the group which, he stated, would be taken to heart for the good of the program.

The final session was devoted to integration plans and several groups met separately to discuss collection and distribution of data bases; production, distribution and use of tools; and integration of new research into the NGS. Cdr. Sears closed the general session by announcing that David Pallett of NBS would manage the performance evaluation and the data base collection efforts, and that James Hieronymus of NBS would manage the technical integration and transition issues involved in the speech recognition program.

Appreciation is due to the Schlumberger Palo Alto Research for their hospitality and use of their facilities for the conduct of this workshop. Particular thanks is extended to Richard Lyon and Robin Wallace of Schlumberger for their assistance in making necessary arrangements.

The cover layout was created by Peter Gustafson of the SAIC Graphics Department using material from Stephanie Seneff's paper on an application to speaker-independent vowel recognition. The figure illustrates the envelope response of 40 channels for the word "Celebrate," and appears as Figure 6 in her paper. For more information see Seneff's paper included herein.

Lee S. Baumann
Science Applications
   International Corporation
Workshop Organizer

AUTHOR INDEX

SECTION 1

REVIEW OF NEW GENERATION SYSTEM FOR
CONTINUOUS SPEECH RECOGNITION

Status of the C-MU Phonetic Classification System

The Feature Group[1]

Computer Science Department
Carnegie-Mellon University

Abstract

The C-MU phonetic classification system is designed to provide a speaker-independent phonetic transcription of an unknown utterance. The system uses perceptually motivated features of speech to locate and classify phonetic segments. This report describes the current system configuration, the research that is performed to develop and improve the system, and the most recent performance evaluation.

System Overview

The input to the phonetic classification system is a spoken utterance. The output is a lattice of phonetic segments with probabilities assigned to each segment. Begin and end times for each segment are determined by *locator* algorithms. Phonetic label probabilities are assigned to each segment location by *classification* algorithms.

At the present time, the segment lattice is created by four separate modules. Each module is designed to locate and classify a particular set of *target segments*; phonetic segments with common acoustic properties. The four modules are:

- **Stop Module:** Designed to locate and classify stop and affricate consonants before sonorants (vowels, liquids, nasals and glides).
  The target segments include [b, d, g, p, t, k, ch, jh] and the stop-like allophone of /dh/.

- **Fricative Module:** Designed to locate and classify phonetic segments that are accompanied by frication noise.
  The target segments include [s, sh, z, zh, f, th, ch, jh].

- **Closure Module:** Designed to locate and classify closures, background noise and pauses.
  Target segments include voiced closures, voiceless closures, epinthetic closures, background noise and pauses.

- **Sonorant Module:** Designed to locate and classify all voiced segments.
  Targets include vowels, dipthongs, liquids, glides nasals and flaps.

Research Process

This section provides an overview of the research process for (a) developing locators, (b) developing classifiers, and (c) analyzing and improving classifier performance. Those familiar with the research process should proceed to the following section, which describes the four modules.

Approach

The general approach is to use perceptually motivated features of speech to make phonetic decisions. This process requires examining speech spectrograms and other visual displays to determine the features of speech that are needed to discriminate among a given set of segments. Feature measurement algorithms are then developed and the resulting feature measurement values are combined to make phonetic decisions.

There are two major advantages to this approach. First, decisions about phonetic segments are based on all available information. It is well known that the perceptual cues for phonetic segments are distributed across both frequency and time, and that perceivers make use of all available information. For example, cues for inter-vocalic [t] vs. [k] may include features in the preceding vowel, features in the closure interval, features in the burst and aspiration and features in the subsequent vowel. The current approach allows us to measure and combine these features to make phonetic decisions.

A second advantage of the approach is the ability to understand and correct errors. By studying pictures of segments that are incorrectly classified, it is possible to understand why misclassifications occur and what can be done to eliminate them.

Development of Location Algorithms

The location algorithms use rules to hypothesize the location of target segments. Location algorithms are evaluated in terms of (a) the percentage of target segments correctly detected in a database of hand-labeled speech, (b) the accuracy with which the left and right boundaries are located, and (c) the number of additional firings produced by the algorithm (an additional firing is any locator firing that does not correspond to a target segment).

The goal of the research is to develop location algorithms that accurately locate each target allophone. The process proceeds as follows:

1. Create pictures of the target allophones showing many different parameters.

2. Select the best parameter(s) for locating the allophones.

3. Develop an algorithm to locate every target allophone in a training database of hand-labeled speech.

4. Evaluate the algorithm in terms of hits, misses, accuracy of boundary location and number of additional firings.

5. Make pictures showing the behavior of the algorithm.

6. Refine the algorithm to eliminate misses, improve boundary location and reduce the number of additional firings.

7. Repeat this process until satisfactory results are achieved on the training database.

8. Test on a new database.

9. Iterate steps 4 through 8.

Development of Classification Algorithms

Classification algorithms use perceptually motivated features and multivariate classifiers to separate additional firings from the target segments, and to discriminate among the target segments. Classification algorithms are evaluated by comparing the segment probabilities produced by the classifier to a database of hand-labeled speech.

The process of building a classifier proceeds through the following stages:

1. Make a picture of each firing of the location algorithm.

2. Sort the pictures into different piles based on acoustic features. For example, pictures of locator firings for each occurrence of [t] may be grouped into separate piles for aspirated [t]s, [t] in [st] clusters, and unaspirated [t]s before unstressed vowels. "Extra firings" are also grouped into categories based on common acoustic features.

3. The locator firings are now given new labels; e.g., "aspirated t", "cluster t", "short t".

4. The pictures are studied to determine the features needed to discriminate among the category labels.

5. Feature measurement algorithms are developed and feature measurement values are collected for each category label.

6. A classifier is built that provides the best possible discrimination of the labeled categories.

7. The classifier is tested on a new database of hand-labeled speech.

Classifiers are created using an interactive graphics program called "Classgraph." Classgraph allows the researcher to design a classifier that incorporates knowledge about how features interact to define phonetic categories. Classgraph is a tool for both understanding the feature relationships in a particular classification and for creating the classifier structure and boundaries. It allows the user to look at hand-labeled training data in two-dimensional projections of the original feature space. These projections can be selected by hand or computed automatically.

The user may create *regions* in Classgraph by drawing a set of boundaries. Additional regions may be created within the new region, or in the region excluded by the original region, allowing the user to create a tree structure of regions. This process is guided by the user's knowledge of how the decisions should be structured and how the projections should be made at each node of the decision tree.

The goal when using Classgraph is to create regions containing the correct proportion of category labels given the available featural information. Thus if the featural information shown on the visual displays is sufficient to uniquely identify the segment [s], we attempt to use these same features to create a region in Classgraph containing only labeled samples of [s]. Similarly, if the features are sufficient to determine that the segment was either [s] or [th] but not [f], then the feature measurement values are used to create a region that contains labeled samples of [s] and [th], but no labeled samples of [f].

Error Analysis

The classifier is then tested on a new set of utterances to detect misclassifications. Misclassifications are eliminated using the following procedure:

1. Determine the regions in Classgraph to which each misclassified segment was assigned.

2. Analyze pictures of each type of misclassified segment in each region.

3. Compare the misclassified segments to other segments in the region. Ask the following questions:

   • What features were supposed to discriminate the misclassified segment from the segments in the region?

   • Why did they fail to do so?

   • What additional features are needed to redirect the misclassified segments to the appropriate region?

4. Refine the existing feature measurement algorithms or develop new feature measurement algorithms.

5. Compute measurement values.

6. Create a new Classgraph structure that eliminates the misclassifications.

7. Run the system on a new set of utterances.

8. Continue this process until the classifier performs in an acceptable manner.

The Modules

Before describing the modules, a few general points should be made. First, algorithms in all modules were designed to look no more than 250 msec ahead of the current time frame. This constraint was imposed so that the modules could be implemented to perform in near real time. Second, at the present time, the four modules operate independently. In future iterations, the modules will share information. Finally, in each module, the locator algorithm produces *additional firings*; that is, segments are hypothesized that do not correspond to the target segments. Additional firings fall into two categories. "Bad firings" are spurious hypotheses that are not useful for word recognition. "Good firings" correspond to phonetic categories (other than the target categories) and may provide useful information to other modules in the system. For example, the closure module locates many of the intervocalic weak fricatives that are not found by the fricative module. We use features and classifiers to eliminate bad firings from further consideration and to correctly classify good firings.

Stops

The Stop Module is designed to locate and classify [b], [d], [g], [p], [t], [k], [ch], [jh] and stop-like [dh] before any sonorant (vowels, nasals, liquids, and glides). The components of the module are (a) a locator algorithm that finds stop bursts and sonorant onsets and hypothesizes segments between the two, (b) a classifier that discriminates among target segments and bad firings, and (c) a classifier that discriminates among the target segments.

The stop locator detects stop bursts followed by sonorant onsets. A stop burst is defined as any jump in high frequency energy greater than a threshold value. Sonorant onsets are detected by finding significant changes in low frequency peak to peak amplitude following each possible stop burst. Target segments are then hypothesized between each stop burst and each sonorant onset within 200 msec. Target segments that do not have a preceding closure are eliminated from further consideration.

Next, a classifier eliminates bad firings. Bad firings include categories such as "better burst later," "better burst earlier," "no real sonorant present," and so forth. This classifier was designed to eliminate bad firings without losing any target segments.

A final classifier assigns a phonetic label probability to the remaining segments. The first stage of this classifier assigns each segment to one of seven categories based on two features: The duration of the segment and the average high frequency energy in the segment. Finer distinctions are then performed using features appropriate to the individual segments. At this time, only the obvious features have been used to perform fine phonetic distinctions, such as the spectral properties at burst onset, spectral center of gravity, and relative amplitude differences. More sophisticated features (such as formant transitions in the adjacent sonorant) will be included in the next iteration of the module.

Fricatives

The Fricative module is designed to locate and classify the fricatives [s], [sh], [z], [zh], [f], [th] and the affricates [ch] and [jh] in any context. The module consists of (a) a locator algorithm, and (b) a classifier that discriminates among target segments and additional firings.

The locator algorithm uses zero crossings and low frequency peak-to-peak amplitude to find fricatives. Local maxima in zero crossings of the waveform trigger segment hypotheses. Segment boundaries are determined by a significant drop in zero crossings or a significant increase in low frequency peak to peak amplitude.

Segments are then classified as targets or additional firings. Additional firings include aspirated stop consonants, unaspirated stops, syllable-final stop release bursts, and weak voiced fricatives ([dh] and [v]). The main features used in classification include spectral center of gravity, presence of voicing during or preceding the segment, zero crossings, and amplitude onset characteristics. At present, contextually important features (such as the spectral broadening caused by a subsequent [l] or [r]) have not yet been incorporated into the classifier.

Closures

The closure module locates and classifies closure and nonspeech intervals based on dips in two parameters; energy and overall peak to peak amplitude computed from the waveform. The threshold values used to postulate closures and nonspeech intervals are normalized to the maximum energy and peak to peak values observed thus far in the utterance so that hypothesized segments are based on relative changes.

Locator firings are classified as targets or additional firings. Targets include voiced, voiceless and epinthetic closures, pauses and nonspeech intervals. Additional firings are classified as weak fricative, weak sonorant, aspiration, or glottalization.

The classifier was developed one category at a time, starting with the target categories (voiceless closures, voiced closures, pauses). The main features used to classify target categories are pitch within the segment (implies voiced closure), zero crossings, high frequency energy and low frequency peak to peak amplitude. The main features used to classify glottalized additional firings are the standard deviation of the low frequency energy and the presence of low peak to peak "humps." The main features used to classify weak fricatives are zero crossing and high frequency energy.

Sonorants

The sonorant module locates and classifies all voiced sections of speech including vowels, nasals, liquids, glides, flaps, voiced closures, and voiced fricatives.

Sonorant regions are located using low frequency (0 - 700 Hz) peak to peak amplitude. The algorithm is designed to find all sonorant stretches of speech, and therefore includes voiced fricatives and other phonetic events. Within each voiced region,

possible segmentation events are found. These events are: dips in low frequency amplitude, changes in low frequency amplitude, and changes in the ratio of mid frequency (500 - 3500 Hz) amplitude to low frequency amplitude. Each possible segmentation event is classified into one of three classes: not a segment boundary, maybe a segment boundary, and definitely a segment boundary. A lattice of possible segments is created using rules about the combination of these classified segmentation events.

The segments in the lattice are then classified. A separate classifier was developed for each sonorant label. In addition, a classifier was developed for each type of additional firing. Each of these classifiers discriminates between the target label and all other firings. The classifier probabilities are combined by taking the probability of each label from it's classifier and renormalizing to make the probabilities of all labels sum to one.

The features used include formant frequencies and trajectories, duration, relative amplitudes in various frequency bands, and spectral center of gravity in different frequency bands.

Evaluation

Evaluations were performed for each module, and for the combined output of the modules. The evaluations were performed on 100 sentences produced by 10 speakers: 50 phonetically balanced sentences and 50 sentences from an Electronic Mail task. These sentences were not used to train the modules.

The 100 test sentences were hand-labeled phonetically. Modules were evaluated by comparing the locator/classifier output to each occurrence of a hand-labeled target segment. The following tabulations were made: (a) percentage of targets located (hits), (b) percentage of targets missed, (c) left and right boundary alignment, (d) rank order of the target label provided by the classifier and (d) additional firings.

A target was scored as a hit if the hypothesized segment overlapped the hand-labeled target. If more than one segment was hypothesized for a target, the hypothesized segment with the best boundary alignment was selected as the correct firing. The remaining hypotheses were classified as "additional firings". If no hypothesized segment overlapped the hand-labeled target, the target was missed. Classification performance was evaluated in terms of the rank order of the target label provided by the classifier.

Stop Module

The stop module located 96% of the 314 targets in the 100 sentences. The majority of the missed segments were voiced stops preceded by nasals. In this context, there is minimal closure information; the closure was not detected and the segment was rejected as a stop. 99% of segments located were within 20 msec of the hand-labeled boundaries.

Of the 301 targets that were located, 54% were correctly classified as the top choice, with 77% in the top three choices.

There were 247 additional firings: 0.8 for each hit. 69% of these firings corresponded to stop-like events (glottalized vowel onsets or fricatives following stop closures) while 31% were judged to be "bad" firings. More work is needed to correctly classify these additional firings.

Fricative Module

The fricative module located 96% of the 361 targets in the 100 sentences. All of the missed fricatives occurred in fricative-fricative or affricative-fricative contexts (e.g., the [f] was missed in "messages from"). 98% of the located segments were within 20 msec of the hand-labeled boundaries; for those segments with bad boundaries, the average left-boundary error was 40 msec and the average right-boundary error was 80 msec.

Of the 346 fricatives found, 51% were correctly classified as the top choice, with 74% in the top three choices.

There were 326 additional firings: 0.95 for each hit. Additional firings were classified as stop, aspirated stop burst, dh, v, and h. 78% of the additional firings were correctly classified.[2]

Closure Module

The closure module located 98% of the 628 targets in the 100 sentences. 95% of all segments located were within 20 msec of the hand-labeled boundaries; for those segments with bad boundaries, the average left-boundary error was 80 msec and the average right-boundary error was 55 msec.

Of the 613 closures that were located, 95% were correctly classified as the top choice, with 99.5% in the top three choices.

There were 408 additional firings: 0.65 for each hit. Additional firings were classified as aspiration, weak fricative, glottalization, v or weak sonorant. 93% of the additional firings were correctly classified.

Sonorant Module

The sonorant module located 98% of the 1209 targets in the 100 sentences. 88% of segments located were within 20 msec of the hand-labeled boundaries; for those segments with bad boundaries, the average left-boundary error was 65 msec and the average right-boundary error was 55 msec.

Of the 1179 sonorants correctly found, 56% were correctly classified as the top choice, with 79% in the top three choices.

There were 2283 additional firings: 1.9 for each hit. 97% of the additional firings were correctly classified. We also scored the above additional firings for those occurring within a segment and those spanning two or more segments; 90% of these additional firings occurred within a segment.

Coverage Across Modules

The above evaluation examined the percentage of target segments located and classified by each module. However, this evaluation is incomplete for two reasons:

1. A target segment missed by one module may be located and correctly classified as an additional firing by another module.

2. A segment not targeted by any module may be located and correctly classified as an additional firing by another module. For example, flaps were not targeted by any module, but were classified as additional firings by the sonorant module.

---

[2] An additional firing was considered to be correctly classified if a label in the lattice corresponded to the broad class of the hand-labeled segment

Twenty sentences were analyzed to determine the amount of coverage provided by the combined output of the four modules. This analysis yielded 604 hand-labeled segments. Of the 604 segments, 97% were located and correctly classified by one of the four modules. The missed segments were distributed as follows:

- 1% of the 604 segments were targets missed by all modules.

- 2% of the 604 segments were not targets, and were not found by any module.

Reading Experiment

An experiment was also performed to test the "readability" of the phonetic segment lattice. The question asked in this experiment was: "How well can a person who is familiar with the Electronic Mail task (i.e., the vocabulary and grammar) read sentences from this task given only a segment lattice generated by the phonetic classification system?"

In the experiment, a member of the research team (JW) was presented with segment lattices for 17 different sentences. The sentences were created for the experiment and were unknown to the subject. The sentences were selected from a set of twenty sentences recorded by 10 male and 10 female speakers (3 sentences were used for practice). For each sentence, the subject was presented with the segment lattice produced by the four modules and a list of the 243 vocabulary words used in the Electronic Mail task.

Of the 100 words in the 17 sentences, only two were missed. Both errors consisted of confusions between the words "two" and "ten". This result suggests that the current system output is nearly sufficient to recognize words from continuous speech in a highly constrained task when higher level constraints can be used as efficiently as a human problem solver.

Assessment

All of the modules need a great deal of improvement. Only stop consonants are located well and only closures are classified well. It is likely that classification performance must be improved by at least 15% to 20% to achieve acceptable word level accuracy for a vocabulary of 1000 words.

Fortunately, there are many obvious ways to improve the performance of the locators and classifiers. The research process allows us to understand misclassifications, develop the appropriate feature measurement algorithms and improve classification. We therefore hope to achieve significant improvement in classification performance in subsequent iterations of each module.

Building Parallel
Speech Recognition Systems
with the Agora Environment

R. Bisiani, A. Forin

Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA 15213
[412] 268-3072

Abstract

Agora is an environment that supports the construction of large, loosely-structured programs that manipulate complex data structures, e.g. knowledge based systems. Agora can be customized to support the programming model that is more suitable for a given application. Agora has been designed explicitly to support multiple languages and highly parallel computations by means of memory caching and pattern directed invocation. Systems built with Agora can be executed on a number of general purpose multiprocessor architectures.

1. Introduction

Our long-term goal is to develop a software environment that meets the need of application specialists to build and evaluate their own parallel processing systems quickly and efficiently. To this effect we are developing a set of tools called *Agora* (marketplace) that can be used to implement custom environments (called frameworks) for describing, executing, and evaluating parallel systems.

Numerous existing language extensions and programming environments provide abstractions tailored to the incremental design and implementation of large systems, e.g. LOOPS [8], STROBE [9]. Other language extensions deal with general purpose parallel processing, e.g. Multilisp [6], LINDA [4]. Still others deal with the needs of intelligent and heterogenous systems design, e.g. ABE [5]. These languages and environments have different characteristics and capabilities because they have different goals. Agora shares some but not all of their characteristics:

- Agora is a complete environment, not just a language.
- Agora does not define a new language but rather extends the capabilities of existing languages.
- Agora can be used to build heterogenous systems on heterogenous machines.

Agora is not an "environment in search of an application" but is "driven" by the requirement of supporting the implementation of the CMU distributed speech recognition system [3]. During the past year, we designed and implemented an initial version of Agora and successfully used it to build two prototype speech-recognition systems. Our experience with this initial version of Agora convinced us that, when building parallel systems, the effort invested to obtain a quality software environment pays off manyfold in productivity. Agora has reduced the time to assemble a complex parallel system and run it on a multiprocessor from more than a man-year to about one man-month. The main reason for this has been that the interfacing between user programs has been taken care by Agora. Application research, however, calls for still greater improvement. Significant progress in evaluating parallel task decompositions, in CMU's continuous speech project, for example, will ultimately require that a single person can assemble and run a complete system within one day.

Agora's relationship to user code and operating system functions can be explained by using an "onion skin" model: each successively higher layer provides increasingly sophisticated abstractions and tools, relying on the layer below for functions it requires. Figure 1 shows how Agora lies above Mach's facilities [1] and below application-specific abstractions. The Mach layers provide three major abstractions: message passing, shared memory and "threads". *Message passing* is the main communication mechanism: all Agora implementations can run on machines that provide message passing as the only communication mechanism. *Shared memory* (when availble in the underlying computer system) is used to improve performance. *Threads* (processes that share the address space with other processes) are used to support the fast creation of processes (a useful but not always necessary characteristic).

The Agora layers correspond to different needs:

- *The framework layer* is the level at which most of the application researchers will program. A framework is like a specialized environment that has been built to interact with the user in familiar terms. The description, assembly, debugging and production run of an application system are all performed through the same framework.
- *The agent layer* represents the "assembly language level" of Agora. Since all the details about the control and data structure of the system are available at this level, parallelism can be expressed in a convenient way. Although systems can be fully described at the agent level, this level is best used to describe "frameworks" rather than to program user computations. Computations expressed at this level are machine independent.

6

-the cluster layer maps the "agent layer" into a specific computer system. It is in this layer that the most suitable Mach primitives are selected, the code is compiled and linked, tasks assigned to machines, etc.

While Agora can offer general support for constructing parallel systems, our work tracks closely the particular needs of CMU's continuous speech recognition system. In Figure 1 and throughout our discussion, we frequently illustrate Agora features by referring to the speech system. To introduce Agora's role in developing a specialized application we next summarize the development steps for a parallel speech recognition system.

Application engineers first program one or more "frameworks" that implement the computation environments that an application requires. In the case of the continuous speech system we currently have a different framework for each of the major components and a "global" framework that ties the components together (see Figure 2). In the case of Figure 2 the framework provides data flow and "remote procedure call" communication mechanisms. A framework provides all the tools to generate and maintain *framework instantiations*, i.e. frameworks with user provided code and data.

Researchers can then use frameworks to create framework instantiations. An instantiation of the framework is generated by describing the computations (the blocks in Figure 2), the data (the arcs in Figure 2) and the communication (the interconnections between nodes and arcs). Components of a framework instantiation can themselves be instantiations of some

other framework. In the speech system, for example, the word hypothesizer is described by using a framework that embodies the asynchronous control necessary to run the word hypothesizer in parallel together with code to display the data processed; a user need only be familiar with the algorithms and the language in which they are written. Researchers can then instruct the framework to map agents (via resource-sharing *clusters*) to processors. The framework, under user control, can then execute one or more system agents while the user can inspect the elements generated by means of an *element editor*. Finally, a framework can be instructed to run the complete system.

Frameworks are described in terms of C or Lisp procedures (*called agents*) that communicate with the rest of the system only through streams of typed data elements (called element streams, or ES) whose type is globally defined. An element might contain, for example, information postulating candidate phonemes and their likelihoods for a given time interval. The code provided by the application builder can also be in either C or Lisp.

This paper is an introduction to some of the ideas underlying Agora. The current design of Agora is the result of the experience acquired with two designs and implementations carried out during 1985. One of these implementations is currently used to support the execution of a prototype speech recognition system on a network of Perqs and microVaxes. The design described in this paper is expected to be running on a shared memory multiprocessor by the end of the second quarter of 1986.
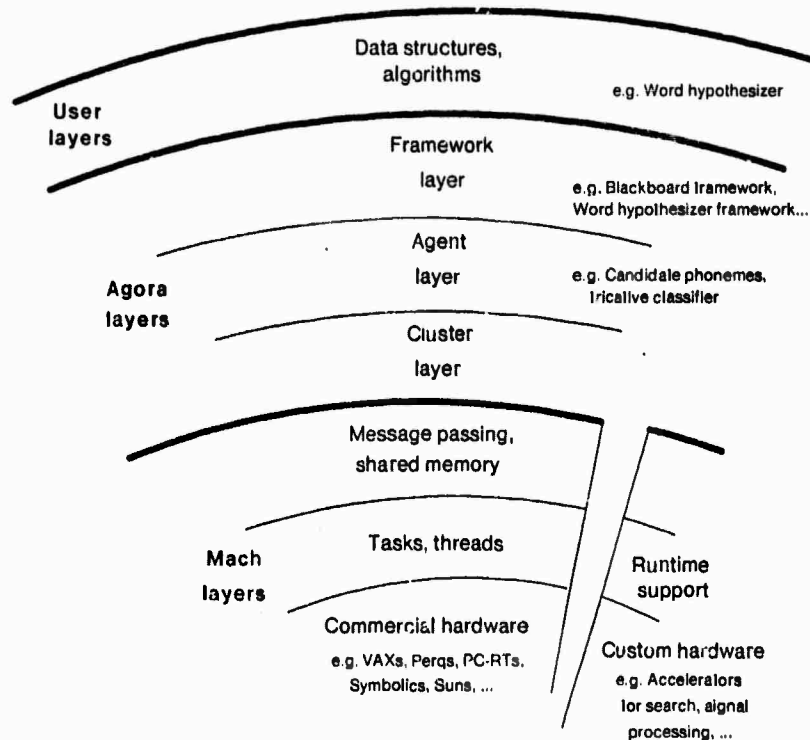
Figure 1: Onion-skin Model of Agora and its Interfaces

2. The Framework Level

The framework level is the primary interface for describing, starting up and running an application. Different applications require different computational models and a different style of interaction with the user. A framework unifies all the necessary tools within a single environment so that they can all benefit from knowledge of the task. Most of this knowledge is built into a framework by its designer, e.g. by adding a display procedure.

Agora contains a number of tools that can be used when building a framework. Each tool can be parametrized to fit the requirements of the framework. Tools are controlled through a common graphical user interface. Agora contains tools to support:

- **Task description.** The systems that we envision building with Agora contain up to 50 independent computations). Systems of this size contain a large number of interconnections and are complex enough to make it impossible to check the correctness of the description by hand. Agora provides a graphic structured-editor that performs a number of checks on the correctness of the system structure. This editor can be parametrized to fit a given framework as a VLSI editor can be parametrized to fit a given technology. This editor also contains functions to start up a system and perform some checks of the structure of the system at run time.
- **Custom display of user data.** When debugging an application it is useful to examine the data that flow within the system in a compact, graphic form. Agora provides a number of facilities that let a framework builder specify "display functions" and then attach them to data communication paths.

- **Editing of elements.** Streams of data can be generated from files and edited interactively by means of a "stream editor". The editor is automatically configured by the information provided by the task description editor. The user can read hypotheses from files or from streams, write hypotheses to files or insert them into streams, or just browse through the elements.
- **Multiprocess debugging.** Agora contains a debugger that continuously traces memory activity in a non-obtrusive way, performs controlled replay of a particular stream of agent activations and can start a number of element stream and language debuggers in parallel on multiple element streams and agents.
- **Performance monitoring.** Agora provides performance statistics that can be displayed in real time or sent to user code to be used as the input of load balancing procedures.

2.1. Example

We will use as example the description of a realistic (although simplified) component of the CMU speech recognition system: a word hypothesizer that computes word hypotheses "anchored" at specific times in the utterance.

At the framework level this component is characterized by three data types: *anchors* that specify when a word matching should occur, *phonetic hypotheses* that are used to perform the match and *word hypotheses* that are the result of the match. Two functions are required: *the matching function* that hypothesizes words from phonemes and *the condition function* that checks if
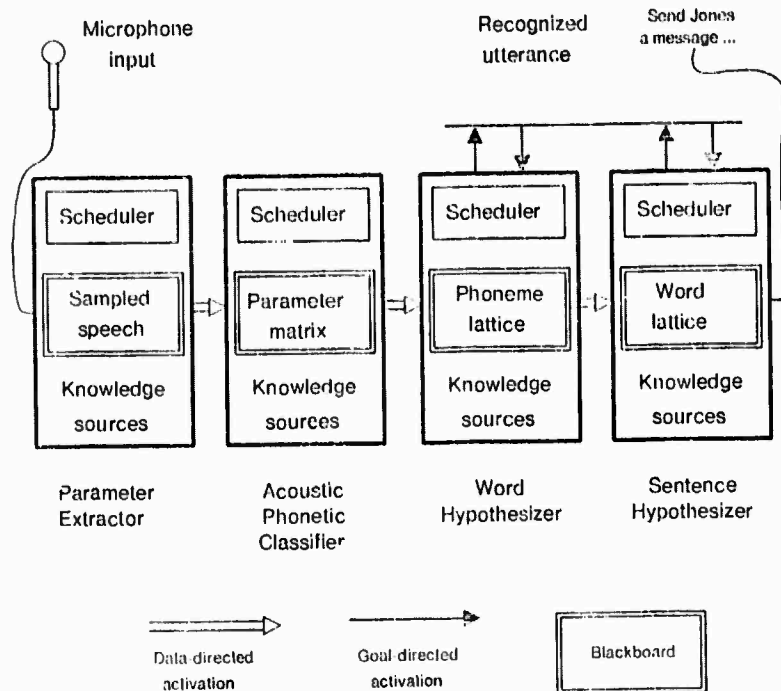
Figure 2: Pictorial Representation of the Distributed
Speech System Framework

there are enough phonemes within a time interval from the anchor. The word hypothesizer must be able to receive anchors and phonemes in any order and perform a word matching around each anchor after having checked that all the necessary phonetic hypotheses are available.

The word-hypothesizer framework lets a user specify the two functions and binds them with the agent level description that provides the parallel implementation (see Section). The framework also contains a display function that can be altered by a user.

3. The Agent Level

Agora (as any other environment) cannot provide optimum performance independently of the style of computation that a task requires and independently of the architecture and implementation of the computer system on which it runs. Agora's agent level is particularly suitable to tasks where the processing that is required is almost but not completely known in advance and some latency in setting-up a computation can be tolerated. Agora also has explicit support for pipeline and parallel/pipeline computations.

These are the main features of Agora:

-Caches. Agora manages on behalf of the user a "global" memory that is organized in streams of records. Dynamically, part of this memory can be "cached" into the address space of a particular computation. Caches:

 *define a convenient model of structured and protected shared memory;
 *can only be accessed through appropriate primitives that guarantee synchronization;
 *can be implemented regardless of the fact that the computer system used has or not a physical shared memory;
 *allow automatic optimization of data distribution.

Analogies can be drawn to the hardware caches in multiprocessors, Agora's model resembles *ownership* caching schemes, see [7].

-Hidden parallelism. Computation descriptions are independent of the fact that parallelism is available in the underlying computer system. In many cases parallelism and pipelining can be automatically generated by the system.

3.1. Main Abstractions
The main abstractions that Agora presents to the framework builder are:

Elements: the (only) form in which data are transported and stored;

Element Streams: sequences of elements;

Agents: the unit of processing that executes concurrently with other agents, exchanges data using element streams and is activated when certain patterns of elements are generated;

We will describe each of these components and explain how they interact.

3.1.1. Elements
Agora is centered around representing data as streams of elements of the same type (element can be regarded as variable-size records). In a speech recognition system, for example, an element could be a phoneme, word, sentence or some other meaningful intermediate representation of speech. Each stream has a name that completely identifies the stream, a type (from a set of globally-defined types) and is ordered by the time of "arrival" of its elements. Any agent that knows the name of a stream can perform operations on it since Agora "registers" the name of each stream when it is created. Since "names" are global the only condition for sharing a stream between agents is that a stream be first "created" by an agent (the agent cache becomes then the "owner" of the stream) and then declared "shared" by another agent. Ownership can be transferred from agent to agent.

Element types are described *within the agent code* by using the syntax of the languages that are used the program the agents with very few additions. This means that users need not learn a different language. The additional information is stripped from the source code by Agora before the code is handed to the compiler or interpreter. This is in contrast with other language-independent data transport mechanisms, like the mechanism described in [2], that use a separate language to define the data. The type declarations can contain extra information for debugging purposes, e.g. the legal values that elements can assume, display procedures, etc.

Agents can refer to subsets of element streams by using *capabilities*. Capabilities are identifiers that contain three pieces of information: the stream name, and two indexes that identify a number of contiguous elements by their first and last element.

Capabilities are manipulated by Agora functions and can be used to "copy" from a cache into the address space of an agent and to copy from the agent space into a cache (often no real copy will be necessary). There are three "modes" of access: *Read-only*: the data cannot be written back into the cache. *Add-element*: data can be added at the end of a stream. *Replace*: elements can be replaced by an equal number of new elements.

3.1.2. Agents
Agora forces the user to split a computation into separate components called agents. Functions (called agent functions) are incapsulated in a module (called agent) that executes concurrently with other agents. Agent functions are completely independent of the topology of the system they are used in and must only be able to deal with the element types that they declare. An agent can contain more than one agent function, each activated by a different pattern (see later). Only one agent function is active at any given time. Examples of agents in a speech system include a vowel classifier in the phonetic classifier cluster and a matcher in the word hypothesizer.

Agents are created by an Agora function (*build*), each call to *build* potentially generates multiple instances (called *clones*) of

the same agent. If an agent is programmed as a set of state-less functions the number of instances does not affect the computation but for the fact that some agents computations might be executed in different order (which, in turn could affect the outcome of the computation).

Agents are associated with a pattern that is checked every time one of the streams mentioned in the pattern changes. Multiple patterns that refer to the same stream are evaluated sequentially in the order they were declared but starting from the last pattern that successfully matched. Only the streams that have been "cached" in the agent in which the pattern is specified can be used within a pattern.

The pattern is expressed in terms of "arrival events" (the fact that an element has entered a stream) and in terms of the values of the element fields. For example, one can specify a pattern that is matched every time a new element enters the stream or that only matches if a field in the element has a specific value. More than one stream can be mentioned in the same pattern but no variables are permitted in the pattern (i.e. there is no binding). It is also possible to specify if an event must be considered "consumed" by a successful match or if it can be used by other patterns (this can be very useful to demultiplex a stream into different agents or to guarantee mutual exclusion when needed).

This is how a "typical" agent will look (the syntax has not been formally specified yet):

```
agent foo(list of capabilities and parameters);

    entry point at creation time:
        <code>
        ......
        creation of new and shared element streams;

        .....
        <code>
    end of agent function;

    entry point triggered by a pattern:

        ......
        <code>
        ......
    end of agent function;

    <more entry points>

end of agent foo;
```

<code> can contain any statement in the language that is being used and any of the following Agora functions, expressed in a way that is compatible with the language used.

build(...) -> agent forks another agent, specifies when the agent should be activated, specifies capabilities to be passed to the agent upon activation, specifies the desired multiplicity of the agent (clones). The agent instantiation(s) remains active and can be re-activated an unlimited number of times. The user can explicitly allow or disallow Agora to "clone" an agent (use the same address space) when it is built more than once.

create(...) -> capability creates an *element stream*(ES) with a global name and a global type. It also specifies the legal operations on the ES.

share(...) -> capability binds a local capability with a previously declared ES and specifies the operations that can be performed on the ES (read, write, read/write) by the local agent. The operations specified must be consistent with the attributes of the stream as specified in the corresponding create( ).

activation( ) -> capability returns a capability to the element(s) that caused an activation. If elements from more than one stream are involved successive calls to *activation* will return all the capabilities.

get(capability) ->local variable copies the elements specified by the capability into the agent address space.

replace(capability, local variable) replaces the elements pointed to by the capability with the same number of elements contained in the local variable.

add(p,capability) adds the data to the element stream pointed to by the capability.

terminate(agent) terminates all the active instances of the agent effectively canceling the effect of build. Can be used by an agent on itself (and its siblings).

update(agent, pattern) changes the pattern matching clause.

regulate(agent, new – power) controls the resource allocation.

3.2. Example
This fragment of code describes the agent level implementation of the work hypothesizer framework. Please note that the use of Agora's constructs and not the syntax or the (C-like) language are important. The time when to compute a word hypothesis is generated elsewhere and added to a stream called *anchor-points*. A word-hypothesization can only be performed if all the acoustic-phonetic hypotheses within *delta* time units from anchor are available. Phonetic hypotheses arrive at unpredictable time and in any order.

First, a few (six) agents that wait for "anchor points" are created. When any of these agents receives an "anchor" it checks if there are enough phonetic hypotheses and, if so, executes the match function. If not enough hypotheses are available, it creates another agent that will wait for the phonemes.

```
Type anchor {
    AnchorTime : integer;
}
Type phoneme {
    BeginTime, EndTime : integer;
    .....
}
Type word {
    .....
}
/******** ********/
```

10

```
agent setop-word-hyp( )

  anch := create( anchor, "anchor - points" );
  phon := create( phoneme, "phoneme - lattice" );
  word := create( word, "word - lattice");
  bold (word - hypothesize( ),activate each-arrival-of anch,
      6 clones );

end-of-agent;

/*****************/

agent word - hypothesize( )

  phon := share("phoneme - lattice");
  anch := get(activation( ));
/* gets the last element in the stream pointed to by phon and checks
   its time */
  If condition( anch -> AnchorTime, get - last - element( phon ) -> EndTime )
  then { match( anch -> AnchorTime );
      add - element( share("word - lattice"), matched - word ); }
  else bold (wait( anch -> AnchorTime, activate new-arrival-of phon), sharable)
end-of-agent;

/******************/

agent wait( marker : Integer)
  If condition( marker, get - activation( ) -> EndTime )
  then {
    match( marker );
    add - element( share("word - lattice"), matched - word );
    terminate(myself);
  }
end-of-agent;

/******************/

function match ( time - reference : integer , matched - word : word)
  ...
end;

/******************/

function condition( I1,I2 : integer ): boolean
  ...
end;

/******************/
```

4. The Cluster Level

Clusters are collections of agents that can benefit from sharing some of the computer resources. For example, a cluster will contain agents that share an element stream or agents that should be scheduled (i.e. execute) together. An example of a cluster could be a set of agents that accept parameters such as pitch, amplitude, zero crossings and energy levels and produce phonetic units. Agora maintains information on which agents are runnable and on how much of the cluster computation power each agent should be receiving. The "agent power" can be influenced by all the agents in a cluster by using Agora primitives.

Clones (multiple instances of the same agent) have a very effective and simple implementation on the Mach operating system [1] as a single process (task, in Mach terminology) in which multiple "threads" of computation (basically stripped-down sub-processes that share the address space) implement the agents.

Although clusters could also be implemented as Mach tasks, sharing the address space between "random" tasks can be very dangerous. Clusters can be implemented (in decreasing order of efficiency) as multiple processes that share memory or as multiple processes interconnected by messages.

The Agora model of computation provides a variable degree of multiprocessing: agents can be executed by separate processors and clusters can be used to dynamically control the allocation of processors.

One of the characteristics of "intelligent systems" is that the effort expended to find a satisfactory solution can depend on the order in which different activities are scheduled. Agora provides all the components necessary to implement focus-of-attention policies within a system, but the responsibility of designing the control procedures remains with the user. The Agora cluster run-time support procedures match the scheduling requests performed by the agents with the reality of the underlying computer system. For example, if a user indicates that an agent can be replicated and run in parallel, Agora can replicate it and attempt to execute it in parallel with other instantiations of the same agent.

4.1. Example

The mapping of the word hypothesizer framework into a parallel system shows how Mach primitives can be used. The agent *word-hypothesize* will be built as a task if the target machine does not have shared memory. The pragma "6 clones" (in the call to *build*) indicates to the cluster level that the framework builder believes six copies of the agent can be efficiently used. If the machine has shared memory then threads can be used and the pragma becomes irrelevant since new threads can be generated without incurring too much cost. The cluster layer is instructed (by using the pragma "sharable" in the call to *build*) to build the *wait* agents as threads of the same task. This is possible because the agent *wait* and the functions it calls do not use any global data.

5. Conclusions

Agora has a number of characteristics that make it particularly suitable for the development of complex systems in a multiprocessor environment. These include:

- the complexity of parallel processing can be hidden by building "reusable" custom environments that guide a user in describing, debugging and running an application without getting involved in parallel processing programming;
- computations can be expressed in different languages;
- the system topology can be "computed" in real time;
- data are dynamically "cached" to minimize data transfer;
- agents can explicitly and dynamically declare how they interact as far as memory is concerned;
- agents are activated by pattern matching on the element streams;
- agents are described in a way that allows Agora to match the available resources with the requirements of the computation.

11

Main Memory
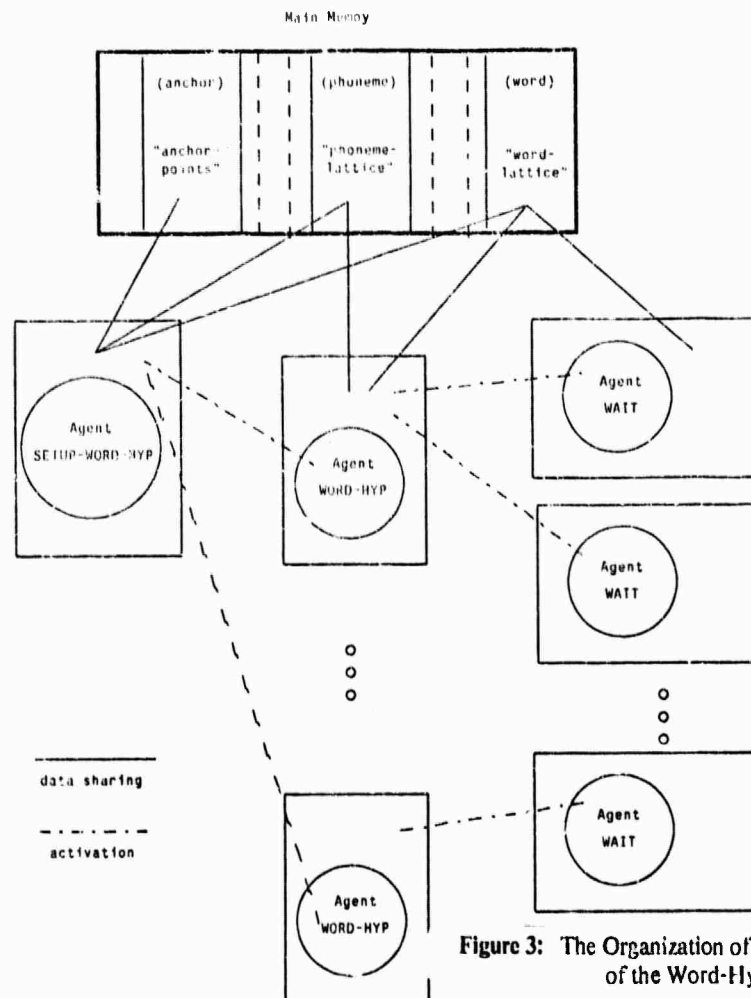
Figure 3: The Organization of the Agent Level of the Word-Hypothesizer

Agora also provides a way of describing the task at hand without any concern for the kind of physical distribution that will be possible on a given computer system. Thus, a system description will not change whether the system is run on a network of workstations, on a shared memory multiprocessor or on a hypercube-network multiprocessor.

Acknowledgements

1. Baron, R., Rashid, R., Siegel, E., Tevanian, A., and Young, M. "Mach-1: An Operating System Environment for Large Scale Multiprocessor Applications". *IEEE Software Special Issue* (July 1985).

2. Birrel, A.D. and Nelson,B.J. "Implementing Remote Procedure Calls". *Trans. Computer Systems 2*, 1 (February 1984), 39-59.

3. Adams, D.A., Bisiani, R. The CMU Distributed Speech Recognition System. Eleventh DARPA Strategic Systems Symposium, Naval Postgraduate School, Monterey, CA, October, 1985.

4. Carriero,N. and Gielernter,D. The S/Net's Linda Kernel. Proceedings of the Tenth ACM Symposium on Operating Systems Principles, December, 1985.

5. Erman,L. et al. ABE: Architectural Overview. Proceedings of the 1985 Distributed Artificial Intelligence Workshop, Sea Ranch, California.

6. Halstead, H. "Multilisp: A Language for Concurrent Symbolic Computation". *ACM Trans. on Programming Languages and Systems 7*, 4 (Oetber 1985), 501-538.

7. Katz, R. H., et. al. Implementing a Cache Consistency Protocol. Twelfth Ann. Symp. on Computer Architecture, ACM and IEEE Computer Society, June, 1985, pp. 276-283.

8. Bobrow,D.G. and Stefik,M.J. A Virtual Machine for Experiments in Knowledge Representation. Xerox Palo Alto Research Center, April, 1982.

9. Smith,R.G. Strobe: Support for Structured Object Knowledge Representation. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, August, 1983.

12

Speech Recognition Experiments
with a Cochlear Model

Richard F. Lyon

Schlumberger Palo Alto Research

3340 Hillview Ave.

Palo Alto, CA 94304

Abstract

There are several ways that a computational model of auditory processing in the cochlea can be applied as the front end of a speech recognition system. For an initial round of experimentation, the fine time structure in the model's output has been used to do spectral sharpening, yielding a "cochleagram" representation analogous to a short-time spectral representation. In later experiments, fine time structure will be exploited for a more detailed characterization of sounds, and for sound separation.

So far, experiments have been done with only two words ("one" and "nine") spoken by 112 talkers, to limit the range of phonetic variation to simple voiced sounds, while providing a good sample of inter-speaker variation. The structure of the vector space of "auditory spectra" has been examined through vector quantization experiments, which yield a measure of information content and local dimensionality.

The inclusion of more dimensions of perceptual variation, such as pitch and loudness, in a speech front end representation is both an opportunity and a problem. Much larger vector quantization codebooks and more training data may be needed to take advantage of the extra information dimensions. A product-code approach and an improved algorithm for finding the nearest neighbor codeword are suggested to help cope with the problem and take advantage of the opportunity.

Preliminary recognition experiments using a single codebook per word and no time sequence information have shown a performance of about 97% correct one/nine discrimination for talkers outside the training set, and 100% correct for second repetitions from talkers in the training set. Further experiments are currently underway.

Introduction

Our experimental cochlear model has been most recently described in terms of its performance on simple "physiology" experiments [1]. Those experiments concentrated on the role of the AGC stages, which serve to partially normalize the output representation in the face of a wide dynamic range of overall amplitude and overall spectrum variations. The dynamics of the gain control process help to preserve perceptually relevant information about loudness and spectrum, emphasizing short-term changes.

The output of the model is regarded as a sequence of vectors in n-space, representing n-channel perceptual spectra. Silence maps to the zero vector, and perceptually louder sounds map to points further from zero. But detailed characterizations of this pattern space are difficult, due partly to its high dimensionality.

The number of important dimensions of variation due to phonetic and talker identity is an important issue in designing recognizers to work in this space, and is discussed in the next section. The following section discusses a set of recognition experiments, including comparisons with LPC. Finally, improved vector quantization techniques to work in this pattern space are suggested in the last section.

The Space of Cochlear Spectra

In the current version of the model, 92 bandpass channels are used to span a range of about 23 barks (about 100 Hz to 10 kHz). By modeling hearing, it is hoped that sounds will map into 92-space in such a way that a simple Euclidean distance in that space will correlate well with perceptual distinctions. Therefore, it is expected that a low-distortion vector quantizer designed to minimize mean squared Euclidean error will preserve most of the relevant information in a cochlear spectra. To explore this notion, codebooks of different sizes and distortions were constructed from various training corpora.

To make codebooks, a modified k-means algorithm was used. In each pass over the training data, new codewords were added to the codebook whenever the distortion to a training vector exceeded a desired distortion bound; at the end of a pass, each codeword was moved to the average of the vectors that were closest to it. Compared to a straight k-means with codebook size doubling, we found convergence to about the same rms distortion for a given codebook size, but in fewer iterations. Having maximum distortion as an independent variable is also useful.

The resulting data on codebook size vs. rms distortion and max distortion for a training corpus of 112 talkers saying "one" and "nine" are shown in Figure 1. The desired value of max distortion, such that reconstructed cochleagrams have clear and continuous formant and pitch tracks, is probably less than the lowest tried so far.
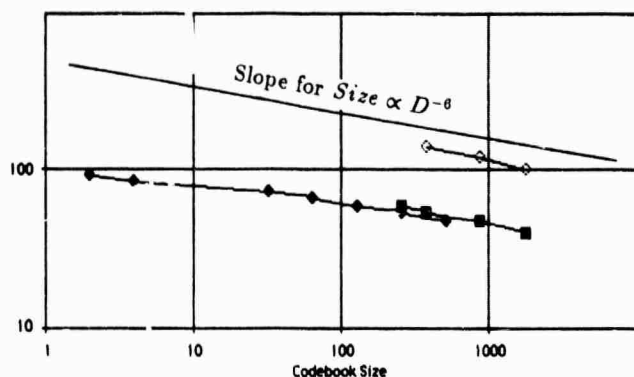
Figure 1: Codebook rms distortion (filled symbols) and maximum distortion (empty symbols) vs. codebook size.

The slope of the size vs. distortion curves (on a log-log plot) should reveal the dimensionality of the subspace that the codewords are packing into. Cutting the distortion by a factor of two will require a factor of sixteen in codebook size increase if there are four dimensions of variation to be covered.

The data show slopes corresponding to about 6 dimensions. Since the phonetic variation in the test corpus is quite small, much of this variation is probably due to talker differences. Since lower pitch harmonics are resolved in the spectrum, and loudness is not completely normalized out, these perceptually important dimensions contribute important dimensions of variation in the data that would not normally be seen in LPC and other common representations.

For the one/nine data, a codebook size of 1801 is barely adequate for high-fidelity coding of cochleagrams of the talkers in the training set. For the complete digit vocabulary, a codebook about five times larger would probably perform similarly. The distortion caused by various codebooks is apparent in figure 2.

Based on these observations, it appears that representing a complete range of phonetic variation (eight or more dimensions), with reasonable fidelity would require a codebook size around 50,000 to 1,000,000. These sizes are far beyond normal practice in the speech recognition field, and require new techniques if they are to be useful.

Recognition Experiments

Since training our existing recognizer [2] to use the cochlear spectrum pattern space will take considerable time, a much simpler test was undertaken first. Using the technique of Shore and Burton [3], a codebook was designed for "one" and another codebook was designed for "nine", using a single repetition of each word from each of the first 5C of the 112 talkers. Setting maximum distortion to 140 for both cases, the codebook for "one" reached a size of 261 and an rms distortion of 45.2, while the codebook for "nine" reached a size of 272 and a 5% higher rms distortion of 47.3.

Recognition proceeded by comparing quantization distortions (rms or total squared distortion) using the two codebooks, without compensation for the different codebook characteristics. No endpoint detection was done, so the generous amount of silence and noise at both ends of the words was included in the distortion measurements.

Testing on the second repetition of the same words from the training talkers led to no errors (in 100 trials). This result is encouraging, since this recognition technique has not previously been very successfully applied to speaker-independent or multi-speaker problems.

Testing on the other 62 talkers showed a serious bias: there were no misrecognitions of "one" as "nine", but ten misrecognitions of "nine" as "one" (5 on first repetition, 5 on second repetition, mostly from different talkers). Overall, on this speaker independent condition, there are 10 errors in 248 trials, or 96% correct. While this does not approach the performance of a good speaker independent isolated digit recognizer on the "one/nine" discrimination task, it is quite respectable for this simple algorithm.

Using order 11 LPC as a parameterization for comparison, with an Itakura distortion measure, we obtained at best 2 errors in 100 trials from talkers in the training set (98% correct), for various codebook sizes, and 14 errors in 248 trials on the other talkers (94.4% correct). Surprisingly, even very small codebooks (2 to 16 codewords) performed well with LPC, so it was decided to go back and try the cochleagrams with small codebooks.

With cochleagrams, it was found that for talkers in the training set, larger codebooks work best (sizes 32 and up gave no errors), but that smaller codebooks do a better job of generalizing to talkers outside the training set (size 32 was optimal with 7 errors in 248 (97.2% correct), while sizes 16 and 64 both were both slightly better than the initial large-codebook experiment, with 9 errors each. These differences may not be significant.

For every codebook size except size 2, the cochleagrams gave fewer errors than the LPC, usually by more than a factor of two.

VQ Algorithm Improvements

In spite of the encouraging results with small codebooks, it seems that to take full advantage of the information in cochleagrams with large talker populations will require very large codebooks. There are (at least) two alternative approaches to making very large vector codebooks practical. First, better fast quantization algorithms can be used to reduce the time cost. Second, codebooks can be constructed as product codes built from a small number of moderate-size codebooks.

Our present quantization algorithm takes advantage of the triangle inequality that applies to the Euclidean distance metric, so that codewords too far from a current best guess need not be examined; this unfortunately requires a table of N^2 inter-codeword distances, and so is impractical

14
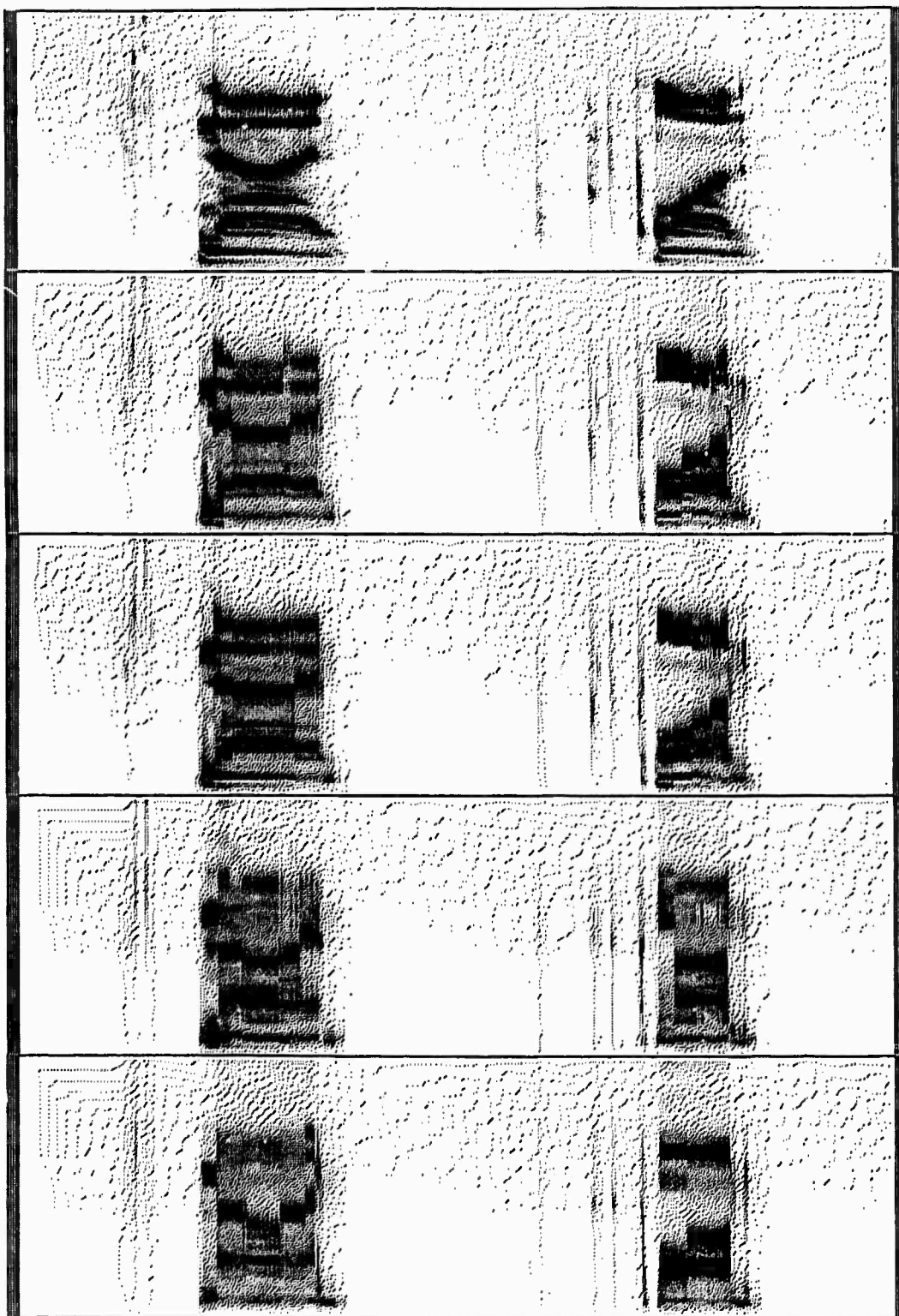
Figure 2: Cochleagram and vector quantized cochleagrams of "nine" and "one". From top: original, size 383 codebook, size 879 codebook, size 32 "nine" codebook, size 32 "one" codebook (talker outside the training set).

15

for much larger codebooks. The FN algorithm [4] uses a tree structure with a branch-and-bound search algorithm to take advantage of the same inequality with less stored information. Another approach which looks promising is to store the dual of the multi-dimensional Voronoi diagram [5] of the code vectors, so that each code vector is linked to its neighbors; in this case, when the current best guess is better than any of the neighbors, no further codewords need be examined. Using the last frame's quantization index as a first guess is very effective in these algorithms. In any case, the auxiliary data structures should be designed such that they are easy to modify when expanding or iterating the codebook.

The product code approach [6] is an alternative way to encode many bits of information per symbol with low distortion and small codebooks. The code space is the direct product of smaller codes, each of which encodes a separate part of the information in the original vector. In the simplest case, the original vector to be encoded is simply split up such that some components (i.e., cochleagram channels) are used as a small vector in one codebook, and the other components are used with one or more other small codebooks. But other vector processing operations could also be used to try to separate the information more cleanly into feature vectors of lower dimensionality. For example, one process could attempt to capture pitch information, another could try to capture first formant information, etc. As long as these "feature extraction" processes don't lose information, the overall vector quantization distortion can be made as low as desired (even if quantizing sub-optimally by independently quantizing with each small codebook). If each feature detecting process captures only one or two important dimensions of variation, the resulting codebooks could be quite small. The structure imposed on the code space by the product code may also be useful in some kinds of recognition algorithms.

Conclusions

The cochlear model produces a spectral representation that captures important dimensions of speech signals. Preliminary experiments show that cochlear spectra lead to about 50% fewer errors in a very simple recognition technique, compared to LPC. Taking full advantage of the extra dimensions of information in cochlear spectra with a wide range of phonetic material and a wide range of talkers may yet require very large vector quantization codebooks or other techniques to extract the relevant features.

References

[1] Richard F. Lyon and Lounette Dyer, "Experiments with a Computational Model of the Cochlea," Proc. *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Tokyo, April 1986.

[2] Marcia A. Bush and Gary E. Kopec, "Evaluation of a Network-Based Isolated Digit Recognizer Using the TI Multi-Dialect Database," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Tampa, March 1985.

[3] J. E. Shore and D. K. Burton, "Discrete Utterance Speech Recognition without Time Alignment," *IEEE Trans. Inform. Theory* **IT-29**, pp. 473–491, July, 1983.

[4] K. Fukunaga and M. M. Narendra, "A Branch and Bound Algorithm for Computing k-nearest Neighbors," IEEE Trans. Computers, **c-24**, pp. 750–753, 1975.

[5] D. T. Lee and Franco P. Preparata, "Computational Geometry—A Survey," IEEE Trans. Computers, **c-33**, pp. 1072–1101, 1984.

[6] John Makhoul, Salim Roucos, and Herbert Gish, "Vector Quantization in Speech Coding," Proc. IEEE, **73**, pp. 1551–1558, 1985.

AN AUDITORY-BASED
SPEECH RECOGNITION STRATEGY:
APPLICATION TO
SPEAKER-INDEPENDENT VOWEL RECOGNITION*

Stephanie Seneff

Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

This paper describes a new system for speech processing that is guided by knowledge about the human auditory system. A bank of critical band filters defines the initial spectral analysis. Filter outputs are subjected to an adaptation model that introduces features such as enhanced onsets and compressed dynamic range. The adaptation outputs are delivered to two parallel channels, each of which produces outputs appropriate for distinct subtasks of the recognition process. One path yields an overall energy measure for each channel, an envelope response that can be identified with "mean rate." The outputs of this path appear useful for locating acoustic events and assigning segments to broad categories.

The extent of dominance of periodicities at each channel's center frequency is captured by a synchrony measure in the other path, which yields a spectral representation with enhanced spectral contrast but with "amplitudes" that are only vaguely related to energy in the corresponding frequency band. The outputs of this stage show clear formant peaks, with smooth transitions over time. These outputs were applied to the task of speaker-independent vowel recognition, using an intermediate "line-formant" representation that is derived by applying techniques similar to those used in vision research. The formant data are first reduced to fuzzy descriptors such as "rising formant over the second half of the vowel centered at 12 Barks." The recognition process involves specifying for each vowel the tolerance ranges for the first two formants, and then searching the list of the line-formants of an unknown token for appropriate matches. Once the line-formants are abstracted, the recognition process is extremely fast, and performance compares favorably with other results reported in the literature for similar tasks.

INTRODUCTION

The human auditory system is an existing speech recognizer with excellent performance If we could build computer models that adequately reflect the transformations that take place in the ear, then the resulting "spectral" representations should be superior to other representations for computer speech recognition. Due to the extensive studies of auditory physiologists, we now know quite a bit about the kinds of transformations that take place, at least at the peripheral level, and it is feasible to build computer models that take these effects into account.

The outputs of such models can often be represented in a spectrogram-like form, such that resonances of the vocal tract show up as peaks at the appropriate frequencies. It may be appropriate to use techniques similar to those used in vision research [7] to abstract from such spectrogram-like representations the relevant information necessary for phoneme identification. By first constructing a "primal sketch" of an auditory-based spectrogram, it is then possible to identify such prominent features as a rapidly falling formant in a form that readily leads to a subsequent straightforward phoneme recognition procedure. Specialized frequency-modulation detectors in the central auditory system [13] lend further credibility to such an approach.

This paper is divided into two major units. The first half focuses on the model for speech processing, which produces two spectrogram-like representations, from which appropriate features could be extracted. The second half is concerned with a proposed new speaker-independent vowel recognition strategy, based on formant trajectories. Features for recognition are abstracted from the synchrony spectrogram obtained as an output of the auditory model. A set of 16 vowels and diphthongs of American English is selected as a small recognition task to serve as a testbed for the proposed method.

PERIPHERAL MODEL

Auditory neurophysiologists have gathered considerable data describing how nerve fibers in the eighth nerve of the mammalian auditory system respond to tone stimuli, tone complexes, and synthetic speech stimuli [1,3,5,6]. From these data it is clear that the ear performs a frequency analysis of auditory stimuli, but that nonlinearities, such as saturation at high stimulus levels, and dynamic effects, such as adaptation, are prevalent in the

measured responses. These data were used for guiding the design of a computer simulation for the peripheral stage of auditory processing.

The analysis system consists of a set of 40 independent channels, which collectively cover the frequency range from 130 to 6400 Hz. A block diagram is given in Figure 1. Each channel consists of a linear critical-band filter, followed by a nonlinear stage (Stage II), intended to capture the prominent features of the transformation from basilar membrane vibration to nerve fiber probabilistic response. The Stage II outputs include the detailed waveshape of the response to individual cycles of the input stimulus; they are still sampled at the original 16 kHz sampling rate. The outputs are delivered to two parallel noninteracting modules. One module determines the envelope response, corresponding to "mean rate response" of auditory neurons. The other module measures to what extent the information near the center frequency (CF) of the linear filter dominates the output; i.e., determines the "synchronous response."

We believe that these two representations are useful for different aspects of the problem of speech recognition. The envelope response tends to show enhanced sharpness of onsets and offsets, relative to the outputs after only the linear stage, and therefore should be useful for determining acoustic boundaries. Furthermore, due in part to saturation phenomena, steady state formants tend to become broader in frequency, which should make it easier to group segments into broad acoustic classes. The synchrony module measures the extent of dominance of information near the filter center frequency in the channel response. This module is described in [10,11]; the only significant modification reported here is the extension of the frequency range from 2700 Hz to 6400 Hz. The outputs of this stage generally show narrow peaks at the formant frequencies, and thus should be suitable for making fine distinctions among within-category competitors.

Filter Bank Design

The filter bank consists of 40 overlapping critical-band filters spanning the frequency region from 130 to 6400 Hz. Their frequency response is shown in Figure 2a, plotted on a linear frequency scale, and in Figure 2b, on a Bark scale [15]. The analog speech is initially filtered at 6.5 kHz cutoff and sampled at 16 kHz. In the interest of efficiency, the filter bank system was designed as a cascade of com-

plex high-frequency zero pairs, with taps after each zero pair to individual tuned resonators. Each individual resonator consists of a double complex pole pair at CF and a double complex zero pair at half CF. The choice of a *double* complex pole pair was motivated in part by observations of phase data on basilar membrane vibrations and nerve fiber responses, suggesting a 2π phase shift through resonance [2,8]. The double zero pair at half CF is necessary in order to produce broad low-frequency tails on the high-CF filters, such as are observed in neural data [6]. A complex zero pair at half CF has been previously proposed by Allen [1].

To specify the critical bandwidth criterion, the frequency scale in Hz was first converted to a Bark scale through a nonlinear mapping function. The conversion was defined by the following set of equations derived by Goldhor [personal communication]:

$$B(f) = \begin{cases} .01f, & 0 \leq f < 500 \\ .007f + 1.5, & 500 \leq f < 1220 \\ 6 \ln f - 32.6, & 1220 \leq f \end{cases} \quad (1)$$

where f is the frequency in Hz, and B is the frequency in Barks. For a given filter, centered at frequency f_0, the critical bandwidth in Hz is obtained by first evaluating $B_0 = B(f_0)$, and then inverting the process to obtain $f(B_0 - 1/2)$ and $f(B_0 + 1/2)$. The difference between these two frequencies is then the critical bandwidth in Hz.
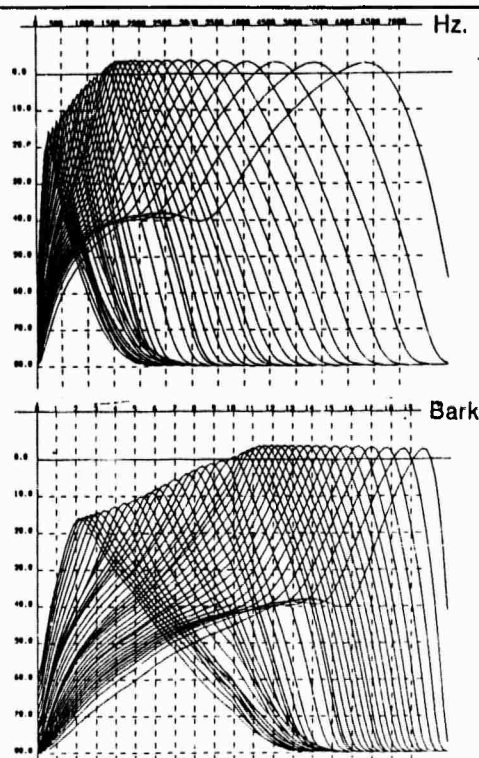
Figure 2: Frequency response characteristics of filter bank.

Figure 1: Block diagram of overall system structure.

18

The major design task was to determine the radius of the double complex pole pair in the s-plane that provides the critical bandwidth, given the existence of the zeros. This was accomplished through linear approximations as follows: [1]

1. Compute the slope, m, of the frequency response of the filter near center frequency (f_c) when all poles and zeros are included except the double pole at $+f_c$, and the two poles at $-f_c$ are assumed to have a radius of 1.0.

2. From a linear approximation to the slope, and using Equation 1 above to determine critical bandwidth, BW, compute two locations above and below f_c where the 3dB points should be. These two frequency locations are separated by a critical bandwidth but may not be equidistant from f_c.

3. By linearizing the unit circle near f_c, use geometric considerations to determine the radius of the poles that will yield critical bandwidth as follows. Let $\gamma(=\sqrt{2})$ be the ratio of the amplitude at the center frequency to the amplitude at the edge of the critical band.

Define:

$$\beta = \frac{\gamma - 1}{|m|\gamma}$$

$$z = \beta + BW/2 - \sqrt{(BW/2)^2 + \beta^2}$$

Then,

$$r = 1.0 - \left|\frac{BW - 2z}{m\gamma}\right|$$

In traditional spectral analysis, speech is typically pre-emphasized prior to Fourier analysis. Some form of pre-emphasis can also be motivated from an auditory standpoint. It has been determined experimentally that broad outer ear resonances should result in a boost in energy above about 1500 Hz by roughly 10 to 20 dB [14]. The gains of the filters in the model are set so as to reflect these resonances, as shown in the figure.

Hair-Cell/Synapse Model

Following the linear filtering stage, each channel is processed independently through a nonlinear stage, to model the transformation from basilar membrane vibration to responses in the eighth nerve. The model incorporates such nonlinearities as dynamic range compression and half-wave rectification, and also captures effects that resemble short-term adaptation and rapid adaptation. No attempt was made to model any long-term adaptation phenomena. The
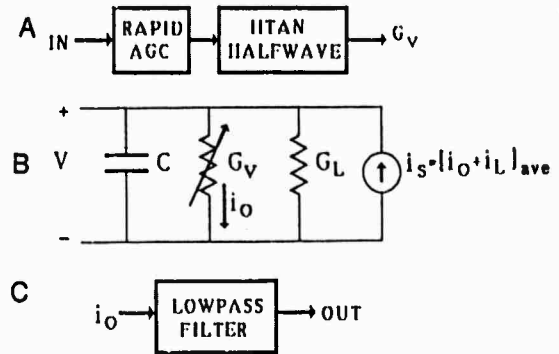
Figure 3: Three-stage hair-cell/synapse model.

output of this stage represents a probability of firing, corresponding to a period histogram.

The model consists of three substages, as shown in Figure 3. The first substage compresses the amplitude at high signal levels and performs a half-wave rectification, based on a raised hyperbolic tangent function. The output of this substage becomes the variable resistance in an adaptation model in the second substage, which is similar to the one proposed by Schroeder and Hall [9]. The "current" through the variable resistance in the membrane becomes the input to the third substage, which is simply a linear lowpass filter to model the partial loss of synchrony with increasing frequency. The shape of this lowpass filter was derived from relevant data obtained by Johnson [5].

In the Schroeder-Hall model, "quanta" of an electrochemical agent are generated at a fixed average rate, r quanta/sec. The probability of firing of an attached nerve fiber is directly proportional to the number of quanta currently in existence and to a permeability function, $p(t)$, that is related to the instantaneous input stimulus level. The quanta are "used up" in direct proportion to the probability of firing, and there is also a certain amount of "leakage," such that a small percentage of the total quanta available disappear without causing a nerve fiber to fire.

Thus, the following equation describes the number of quanta as a function of time:

$$dn(t)/dt = r - [g + p(t)]n(t)$$

where $n(t)$ is the number of quanta at time t, r is the constant quanta generation rate, g is the leakage factor, and $p(t)$ is the permeability function, which depends upon the input stimulus.

The model can be described by an electrical analog as follows:

$$C\frac{dV}{dt} = i_s - [G_L + G_V(t)]V(t)$$

19

where C, a free variable identified with capacitance, can arbitrarily be set to 1.0; i_s, a current source, is the quantum generation rate r; and G_L and $G_V(t)$ are conductances associated with g and $p(t)$ respectively.

The model proposed here is similar to the Schroeder-Hall model, but with one important difference: the quantum generation rate is *not* fixed, but rather adapts so as to try to regenerate on the average the same number of quanta that have been "lost" through the two conductances. Thus $r(t)$ becomes a dependent current source, $i_s(t)$, determined by passing the current flowing through G_L and $G_V(t)$ through a leaky integrator.

The current, $i_0(t)$, through the conductance $G_V(t)$ varies over a wide range with each cycle of the stimulus, whereas $i_s(t)$ tracks only the average value of this current. An effect much like adaptation then occurs in the model because of the delay inherent in the averaging process. Thus the dependent current source becomes the dominant factor in controlling the adaptation rate of the circuit. The time constant of the lowpass filter should therefore be set to a value appropriate for short-term adaptation, i.e., around 30 ms.

Some Examples

Figure 4 shows the outputs of intermediate substages of the hair-cell/synapse model shown in Figure 3, when the input signal consists of a sequence of 2-kHz tone bursts of 50 ms duration, that double in amplitude halfway through, interspersed with equal intervals of silence. The envelope response is apparent on the left, and the detailed waveshape near the cursor is shown on the right. Saturation and half-wave rectification are evident in G_v, whereas the effects of adaptation become apparent in i_0, after substage B. The final output is simply i_0, lowpass filtered.

Figure 5: Responses of 800 Hz channel to first two syllables of "satisfy," spoken by a male speaker, before and after hair-cell/synapse model. Time-expanded response during /æ/ shown on right.

Figure 5 shows the response of the channel tuned to 800 Hz to a segment of natural speech. The response is shown after only the linear filter [top] and after the hair-cell/synapse stage [bottom]. The first formant of the vowel /æ/ is close to the center frequency of the peripheral filter; hence the response is very strong. The time-expanded display on the right shows the detailed shape of the response near the midpoint of the vowel /æ/. One evident effect of saturation is that the periodicity at the formant frequency becomes enhanced relative to the periodicity at the fundamental frequency of voicing. A similar effect is observed in auditory nerve fiber responses to speech-like stimuli [3]. In the weak second syllable, on the other hand, the periodicity at the fundamental is enhanced by the nonlinearities.

Figure 6 illustrates the effect of the hair-cell/synapse stage on the *envelope* response, for a segment of natural speech. Each waveform is the smoothed output of one of the 40 channels as a function of time, with low-frequency channels at the bottom. The original speech waveform and the phonetic transcription are displayed underneath each set, aligned in time. The left panel shows the log magnitude of the smoothed channel outputs, after only the linear filtering stage, hereafter referred to as "Stage I
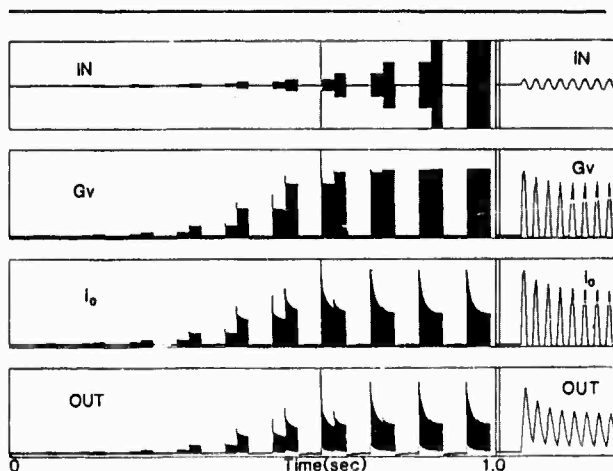

Figure 4: Responses at intermediate substages of hair-cell/synapse model to 2-kHz tone with varying amplitude. Time-expanded response at vertical bar shown on right.
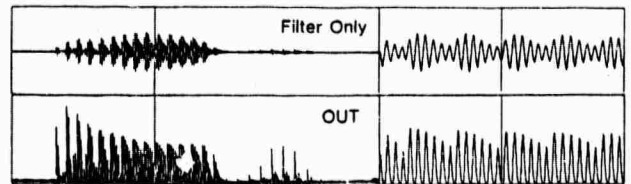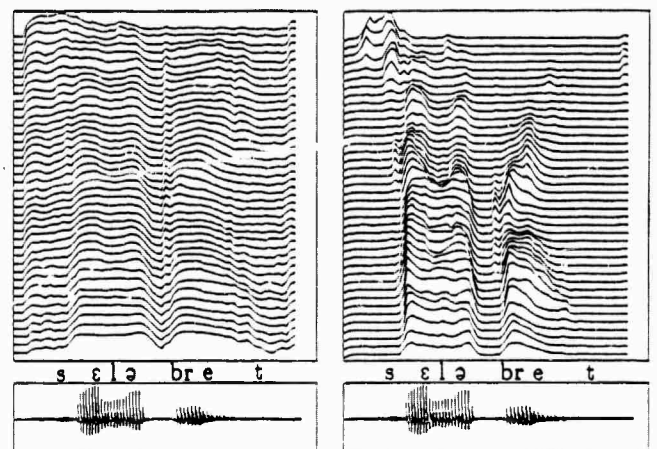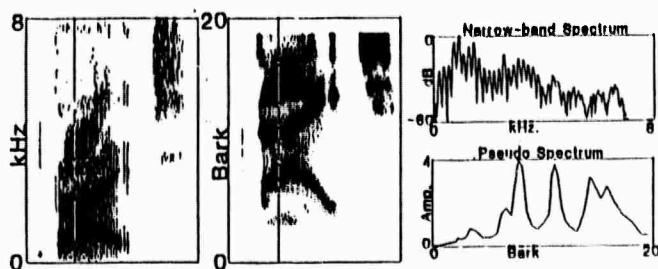

Figure 6: Envelope response of 40 channels for word "celebrate," spoken by a male speaker, after Stage I [left] and after Stage II [right].

20

Figure 7: Left: Wide-band spectrogram for word "bite," spoken by a female speaker.
Middle: Synchrony spectrogram for same word.
Right: Narrow-band spectrum and synchrony spectrum at time of vertical bar.

outputs." The right panel shows the smoothed magnitude response, after Stage II. Acoustic boundaries in time are generally better demarcated *after* Stage II. For example, it is very hard to see the boundaries of the /l/ in the Stage I outputs, whereas these boundaries are much clearer on the right, particularly in the higher frequency filters. Silence intervals, such as the closure for the /b/, show up as a constant value at the spontaneous rate in Stage II, whereas the Stage I outputs during silence or background noise are much less consistent. Rapid formant movements are also better preserved after Stage II, as shown by the rapid rise of F_2 in the /re/ of "-brate."

The Synchrony Spectrogram

The synchrony computation that is performed in the bottom branch of Figure 1 is described in detail in [10] The output of each channel is subjected to an amplitude-normalized scheme for detecting the extent of dominance of energy at the channel's center frequency in the channel output. The frequency resolution is such that the pitch information, in the form of harmonic structure, is lost for male voices but typically retained in the first formant region for female voices. Harmonics between F_1 and F_2 are typically suppressed, because prominent energy at the first formant frequency in the channel output destroys synchrony to the intermediate harmonic. Pitch striations over time are usually absent, due to the amplitude normalization process. Peaks at the formant frequencies are much narrower than in the envelope representation, thus making the synchrony spectrum more suitable for fine distinctions.

The features of the synchrony branch of the system are illustrated in Figure 7. A wide-band spectrogram for the word "bite" is compared with a synchrony spectrogram of the same word, where the latter is displayed on a Bark scale. The synchronous "amplitude" is a highly nonlinear function of the local prominence of a given spectral peak. A narrow-band spectrum in the /a/ portion of the diphthong is compared with a synchrony spectrum on the right.

The synchrony algorithm works surprisingly well in sounds with predominantly high-frequency energy, such as the /t/-burst, because the synchrony measure incorporates energy at d.c. as well as energy at CF. Any strong energy concentration in the signal at high frequencies is mostly converted to d.c. energy, which is passed by the synchrony measure. Prominent peaks in the input waveform well below the CF of high-frequency filters appropriately reduce the synchronous response of such filters, because significant synchrony to the wrong frequency is present.

VOWEL RECOGNITION SYSTEM

After arriving at the speech analysis method outlined above, we then sought to test the applicability of the synchrony outputs to recognition problems. We defined a specific task and developed a complete recognition system for this task, based on the synchrony spectrogram representation. Instead of choosing traditional methods such as dynamic time warping and template matching, we decided to pursue novel alternative methods, motivated in part by vision research. The task is the recognition of the following 16 vowels and diphthongs of English, spoken by multiple native-American speakers, both male and female: /i, e, yu, I, ɛ, æ, a, ɔ, o, ʌ, ʊ, u, aʲ, aᵘ, ɔʲ, ɝ/.

The first step in recognition is to describe spectral peak contours over time in such a way as to conveniently describe formant frequencies and trajectories. The typically two-stage process of (1) formant tracking and (2) abstraction of rates and directions of formant movements is collapsed into a one-step process of directly assigning straight-line segments to the resonance contours in the frequency-time space. The computational procedures are straightforward, leading to a description of the formant information for a given vowel by a list of oriented straight-line segments. These line segments lead naturally to descriptions such as "rising formant," with the slope of the line conveying the degree of rise. No attempt is made to assign the line segments to particular formants, such as F_2. Instead, the recognition process is hypothesis-driven. For each vowel or diphthong to be recognized, a short description of expected ranges of frequency and orientation in the time-frequency dimensions for the first two formants is given.

Line Formant Extraction Process

Figure 8 illustrates the process to obtain a list of straight-line segments describing the formant patterns in a given sonorant segment of speech. The synchrony spectrogram for the word "Burt," spoken by a male speaker, is shown in part (a) of the figure, with the frequency axis represented on a Bark scale. A nonlinear filter-and-quantize procedure defines "On" and "Off" contour regions in time

and frequency, shown in part (b). These correspond approximately to regions where the instantaneous amplitude is greater [On] or less [Off] than the local average.

Each robust peak in a given synchrony spectral cross-section is allowed to vote for a best-fit line segment, restricted to stay within an "On" region and to pass through the time-frequency location of the peak. The selection process, as illustrated in Figure 9, is realized by "drawing" a finite set of straight lines of differing orientations through the sample peak. The average amplitude of all samples on each line, as well as the line's total duration, are determined. The "longest and strongest" line segment is selected as the vote for the given peak. In the figure, only seven different orientations are shown, whereas a total of 11 orientations were used in the system. The votes of the robust peaks are accumulated in a list giving information about the orientation, center-points in time and frequency, duration, and mean amplitude of each line.

The next step is to consider collectively the list of candidate lines over a time interval defined by the unknown vowel's extent. Usually, several peaks vote for the same line or very similar lines. A heuristic algorithm was developed to collapse the list of lines into a new list. "Equivalent" lines are merged into a single representative, and a count of the number of votes being merged is accumulated. Finally, the list is further pruned, and line segments that appear to be insignificant are discarded. Elimination is based on threshold requirements for the number of votes, the minimum allowable duration, and the mean amplitude. In the example, the line segments that remain after pruning are shown in Part (c) of the figure.

The final step in the formant extraction process is to convert the list of line segments into a fuzzy descriptor
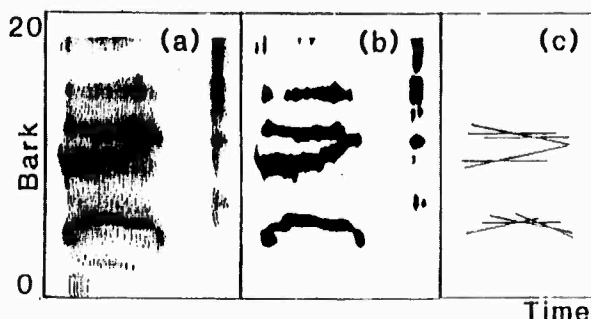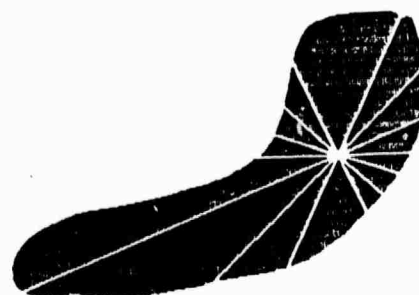
Figure 9: Schematic diagram of process of determining best line candidate to describe possible formant passing through a single peak. Black region denotes "On" contour. Lines at several pre-specified orientations are "drawn" on the synchrony spectrogram, and the strongest (defined by mean amplitude of spectrogram along the line) and longest (where lines are restricted to remain within "On" regions) line segment is selected as the vote for the given peak.

format. The temporal extent of a given line is converted to a verbal description of its extent relative to the vowel end points, such as "first half." Similarly, the strength and orientation of the line are quantized to a small set of possibilities. Only the center frequency is retained as a number. Table 1 lists allowable categories for each item.

Recognition Procedures

The line formant representation was applied in a speaker independent recognition task for the following 16 vowels and diphthongs of English, restricted to /bVt/ context: /i, e, yu, I, ɛ, æ, ɑ, ɔ, o, ʌ, ʊ, u, aʸ, aᵘ, ɔʸ, ɝ/. The only step used for speaker normalization was to reference each line formant's center frequency *in Barks* to an imaginary "zero line" defined as the *median* frequency in Barks of F_0 over the duration of the vowel. F_0 was determined automatically using the method described in [10]. This normalization procedure resembles the method used by Syrdal [12] except that *all* formants, as opposed to only F_1, are defined relative to F_0.

Each vowel candidate is associated with a descriptor list of line-segment specifications for acceptability limits for the first two formants. An example of the descriptor list used for the diphthong /ɔʸ/ is given in Table 2. For /ɔʸ/ to be a candidate solution, the list of line seg-

Figure 8: Illustration of Line-formant Abstraction Process (a) Synchrony spectrogram for word "Burt"; (b) One-bit enhanced spectrogram defining allowable regions for line segments; (c) Resulting line segments describing formants of vowel.

Orientation		Temporal		Strength
Rapid Rise	Rapid Fall	At Start	At End	Strong
Rising	Falling	First Half	Second Half	Medium
Slight Rise	Slight Fall	In Middle	Throughout	Weak
Steady				

Table 1: Categories for descriptors of line formants.

	Freq[Bark]	Orientation	Temporal	Strength
F1	3.1-4.6	*any*	*not* at end	*any*
F2	6.9-10.0	rapid rise, rising	in center	*any*
or	10.1-11.6	rapid rise, rising	second half	*any*

Table 2: Descriptor list for acceptability of diphthong /ɔʸ/.

ments for a given unknown vowel must contain at least one line that matches the descriptor for F_1, and at least one line that matches *one of* the descriptors for F_2. The F_1 descriptor is very general in its requirements, whereas the F_2 descriptors both demand a rising formant over a restricted frequency region, whose range is defined differently depending upon whether the line segment is located in the center of the vowel or toward the latter half.

Each vowel is specified by "approval" requirements for the first two formants, and validity tests for the 16 vowels yield a list of potential candidates. A two-stage pruning process ensues. The first stage is a mandatory elimination of certain candidates if certain other candidates are present. This stage is necessary because the synchrony spectral representation for high-pitched voices may show individual harmonics of the fundamental up to the first formant frequency. In certain cases, a harmonic below F_1 could be an acceptable first formant for /i/, and the underlying F_3 could be an acceptable second formant. If a back vowel such as /ɔ/ also passed its requirements, then the vowel is assuredly not /i/. Rather than require that the /i/ approver look for line segments in a reject region, it is simpler to suppress /i/ on conditions of acceptability of certain back vowels.

The second stage involves a verification step, which may include information about other formants, such as F_3 for /ɝ/. Often the task is to decide between two competing candidates, such as /i/ and /e/. Durational rules may be invoked to make a decision between a short vowel such as /ɛ/ and a long vowel such as /æ/. Such rules include an overlap region where both candidates are allowed to survive.

Recognition Results

The system was trained and tested on a set of 272 tokens, with each vowel spoken once in bVt context by each of nine male and eight female native-American speakers. The system gave a single correct choice 89.6% of the time. A single incorrect choice was given 3% of the time. The remaining 7.4% of the time, there were two choices, one of which was correct. No attempt was made to order these two choices.

Three out of eight errors and nine out of 20 double choices involved the vowels /ɔ/ and /a/. Most of these could be considered as pronunciation errors, for many of the speakers did not make a distinction between these two. Thus if these two are combined into the same class, then 94.7% of the time a single, correct answer is given, 2% of the time a single, incorrect answer is given, and 3.3% of the time two choices are given, one of which is correct. The five errors were /yu/ → /i/, /æ/ → /aᵘ/, /ʌ/ → /ʊ/, /o/ → /ɔ/, and /u/ → /o/.

It is perhaps surprising that the harmonic structure in the F_1 region for female speech did not present a serious problem to the recognition algorithm. While the frequency of F_1 is quantized in such cases to the nearest harmonic of F_0, such quantization is usually adequate to make a correct decision about the vowel. Such structure also necessitated very unrestricted descriptions of allowable orientations for F_1. For example, although F_1 should be falling for the diphthong /aᵘ/, it was not possible to require a falling line segment because of the possible interference of the harmonic structure.

Results based on training data are obviously suspect, particularly in a case like this where hard decisions are made. The system was therefore tested on the 16 words spoken by four new speakers, one male, one female, and two boys, aged 13 and 15. Results are summarized in Table 3, where /a/ and /ɔ/ are collapsed into a single category. Performance for the two teenagers [B1 and B2] was somewhat worse than for the adults, but for the most part vowels for the new speakers fell within the ranges determined from training data. It is significant, however, that after minor rule adjustment, performance for the new data could be improved to the level indicated after the /, without any performance degradation for the training data. Thus, while 16 tokens for each vowel is not enough data to determine the extreme formant positions for those vowels, it requires only minor adjustments of the system to correct errors that occur when new data fall outside the defined ranges.

I.D.	Errors	Two Choices	Count
M1	0/0	0/0	16
F1	1/0	1/0	16
B1 (15 yr)	2/0	2/3	16
B2 (13 yr)	2/1	2/0	16
Total	5/1	5/3	48

Table 3: System recognition results for 16 vowels spoken by four new speakers.
Key: 2/1 → Two errors before and one error after rule modification.

23

SUMMARY AND CONCLUSIONS

It is still premature to know whether an auditory-based speech analysis system will pay off in speech recognition. There are emerging, however, strong indications that auditory based representations are interesting and worthy of further study. We have described here two such representations for the speech signal, one based on mean rate response and the other based on synchronous response. The mean rate response outputs have been used successfully for locating acoustic boundaries and making broad category decisions [4]. Preliminary results using these outputs for syllable detection in continuous speech are encouraging. We are also exploring their utility in fricative identification, and they appear to enhance the differences among the fricative sounds of English, relative to standard Fourier techniques.

We have demonstrated the potential utility of the synchrony spectrogram by means of a speaker-independent vowel recognition task. The method chosen for the portion of the recognizer concerned with extracting features from the auditory representation is similar to strategies used in vision research to cartoonize pictures [7]. By reducing formant information over vowel portions of speech to a small set of fuzzy descriptors, it becomes feasible to require a recognizer to use hard decisions, thus simplifying enormously the computations involved. In spite of the fact that the recognizer makes use of no statistical information and obtains no scores other than "in" or "out," the system has excellent performance on a relatively difficult classification task. The training set used here is not yet adequate to predict the variability of all new data; however, minor rule adjustments could yield improved performance for the test set without changing the original results for the training data. We anticipate that with sufficient training data, it will be possible to capture nearly all of the within-phoneme variability (in this limited bVt context).

The spectral representation, the synchrony spectrogram, chosen for extracting the line formants used for recognition, is particularly well suited to that task, because the formant peaks tend to be well-defined and continuous across time. While it may be feasible to use other representations, such as the LPC spectrogram, as inputs to the line-formant detector, it is not clear that results, either at the level of extracting the lines or at the level of recognizing the phonetic content, would be as good. It is particularly encouraging that the presence of harmonic structure in the first formant region for female speech did not cause problems in vowel recognition.

REFERENCES

[1] Allen, J. B. (1980) "Cochlear Micromechanics: A Physical Model of Transduction," *J. Acoust. Soc. Amer.* 68, 1660–1670.

[2] Allen, J.B. (1983) "Magnitude and Phase-Frequency Response to Single Tones in the Auditory Nerve," *J. Acoust. Soc. Amer.* 73, 2071–2092.

[3] Delgutte, B. (1980) "Representation of Speech-like Sounds in the Discharge Patterns of Auditory-nerve Fibers," *J. Acoust. Soc. Amer.* 68, 843–857.

[4] Glass, J. R., and V. W. Zue (1986) "Recognition of Nasal Consonants in American English," paper 51.5, *Proceedings of ICASSP-86*, Tokyo, Japan.

[5] Johnson, D. H. (1974) "The Response of Single Auditory-Nerve Fibers in the Cat to Single Tones: Synchrony and Average Discharge Rate," Ph.D. Thesis, M.I.T., Cambridge, MA.

[6] Kiang, N. Y-S., T. Watanabe, E. C. Thomas, and L. F. Clark (1965) *Discharge Patterns of Single Fibers in the Cat's Auditory Nerve*, research monograph no. 35, M.I.T. Press, Cambridge, MA.

[7] Nevatia, R., and T. O. Binford (1977) "Description and Recognition of Curved Objects," *Artificial Intelligence* 8, 77–98.

[8] Rhode, W. S. (1971) "Observations of the Vibration of the Basilar Membrane in Squirrel Monkeys Using the Mossbauer Technique," *J. Acoust. Soc. Amer.* 49, 1218–1231.

[9] Schroeder, M. R., and J. L. Hall (1974) "Model for Mechanical to Neural Transduction in the Auditory Receptor," *J. Acoust. Soc. Amer.* 55, 1055–1060.

[10] Seneff, S. (1985) "Pitch and Spectral Analysis of Speech Based on an Auditory Synchrony Model," Ph.D. Thesis, M.I.T., Cambridge, MA.

[11] Seneff, S. (1984) "Pitch and Spectral Estimation of Speech Based on Auditory Synchrony Model," Paper 36.2, Proceedings of ICASSP-84, San Diego, CA.

[12] Syrdal, A. K. (1985) "Aspects of a Model of the Auditory Representation of American English Vowels," *Speech Communication* 4, 121–135.

[13] Vartanyan, I. A. (1975) "Dependence of Responses of Central Auditory Neurons on the Depth and Rate of Frequency Modulation," *Neirofiziologiya* 6, 350–358.

[14] Yost, W. A., and D. W. Nielsen (1977) *Fundamentals of Hearing - An Introduction*, Holt, Rinehart and Winston, New York.

[15] Zwicker, E. (1961) "Subdivision of the Audible Frequency Range into Critical Bands (Frequenzgruppen)," *J. Acoust. Soc. Amer.* 33, 248–249.

RECOGNITION OF
NASAL CONSONANTS IN AMERICAN ENGLISH*

James R. Glass and Victor W. Zue

Department of Electrical Engineering and Computer Science, and
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

This paper presents a progress report on the recognition of nasal consonants, /m, n, ŋ/, in American English. Our present effort is focused on the detection and recognition of nasal consonants using acoustic information in the nasal mur. In Presently, we make no use of nasalization information in adjacent vowel. Our immediate goal is to locate and identify only those nasals in continuous speech that have a clear murmur, leaving the ambiguous ones for future analysis. A nasal *detection* algorithm based on a local decision criterion has been developed, using the outputs of an auditory model. Evaluation of its performance on 365 sentences indicates that 70% of the nasals are correctly located, with one impostor accepted for every nasal. Most of the missed nasals occur in a small number of well defined phonetic environments in which the nasal murmurs are typically articulated poorly. Nasal *identification* is based on a set of five acoustic measures and a strategy that combines a hierarchical decision tree with the likelihood measures for each feature. Evaluation of the classifier performance on the same database indicates that 80% of the nasals and impostors are correctly identified, yielding an overall nasal recognition rate of 56% with an insertion rate of 15%. In order to improve system performance, we must make use of acoustic information from adjacent vowels.

INTRODUCTION

This paper describes our effort in recognizing nasal consonants, /m, n, ŋ/, in American English. Nasal consonants are difficult to recognize for several reasons. First, the characteristics during oral closure, often referred to as the nasal murmur, differ significantly from speaker to speaker because of individual differences in the size and shape of the nasal and sinus cavities. Second, a nasal murmur can be affected drastically by the phonetic environment. In some cases, as in "camp," the nasal murmur is almost entirely absent. In such cases, the detection of the nasal consonant depends almost entirely on the degree of nasalization in the adjacent vowel. Finally, the complex production mechanism makes acoustic characterization of nasals difficult.

The goal of our research is to develop algorithms that recognize nasal consonants in continuous speech. Realizing that in some phonetic environments the presence of a nasal consonant is almost entirely encoded in the adjacent vowel, our strategy utilizes acoustic cues from both the nasal murmur *and* the adjacent vowel. The recognition strategy is as follows. First, regions in the acoustic signal that potentially contain nasal consonants are delineated. Next, acoustic measurements made in each region are used to decide whether the region corresponds to a nasal murmur or an impostor. Measurements are also made in the adjacent sonorant regions to decide whether or not they are nasalized. Finally, the independent decisions based on the consonantal and vocalic portions are combined to provide a single indication of the presence of a nasal consonant.

Our present emphasis is on the detection and recognition of nasal consonants based solely on acoustic information in the nasal murmurs, and not on the degree of nasalization in adjacent vowels. Our immediate goal is to locate and identify only those nasals in continuous speech that have a clear murmur, leaving the ambiguous ones for future analysis. We will first describe a nasal detection algorithm that attempts to delineate regions in the acoustic signal that may contain a nasal murmur. Next we will describe the identification algorithm and evaluate the system performance. We have not yet completed the nasal recognition system outlined above; therefore, this paper should be viewed as a progress report.

NASAL DETECTION

Potential nasal murmur regions in the acoustic signal are detected in several steps. First, the speech signal is automatically delineated into stable acoustic regions. Next, each region is classified into one of four broad phonetic categories, using measurements derived from the gross spectral shape within the region and from the energy contour around a region. Finally, several simple parameters are used to rule out obvious impostors.

Signal Representation

The algorithm for finding stable acoustic regions uses a spectral representation that incorporates known properties of the human auditory system, such as critical-band filtering, half-wave rectification, adaptation, saturation, spontaneous response, and synchrony detection [6]. Specifically, we use the envelope of the output of the filter channels of a hair-cell model, corresponding to the "mean rate response" of the auditory neurons. The model consists of 40 filters equally spaced on a Bark frequency scale, spanning a frequency range from 130 to 6,400 Hz. The hair-cell outputs are represented as a 40-dimensional feature vector, computed once every 5 ms.

We find this representation desirable for several reasons. The hair-cell model tends to enhance the onsets and offsets in the critical-band channel outputs. For low-amplitude sounds, the output corresponds to the spontaneous firing of the neurons, and is greatly attenuated. These two effects combine to sharpen acoustic boundaries in the speech signal. Furthermore, due to the saturation phenomena, formants in the envelope response appear as broad-band peaks, obscuring detailed differences among similar sounds. As a result, we surmise that this representation may be appropriate for broad phonetic classification.

Figure 1 compares the hair-cell envelope response (on a Bark frequency scale) with a wide-band spectrum (on a linear frequency scale) computed during an /s/ (left) and at a /k/ release (right). For these examples, the hair-cell output generally enhances important acoustic cues while suppressing irrelevant information. Note that the outputs of the low-frequency channels of the hair-cell model show only the spontaneous rate response.

Finding Stable Acoustic Regions

Our next objective is to establish stable acoustic regions for further phonetic analysis. Realizing that certain acoustic changes are more significant than others and that the criteria for boundary detection often change as a function of context, we adopted the strategy of measuring the similarity of a given spectral frame to its immediate neighbors. The algorithm moves on a frame-by-frame basis, from left to right, and attempts to associate a given frame with its immediate past or future. Specifically, each frame builds up forward and backward cumulative distance contours $D_F(n, i)$ and $D_B(n, -i)$ respectively, with $D(n, i)$ defined as:

$$D(n, i) = \sum_{j=0}^{i} d(n, j)$$

where $d(n, j)$ denotes the Euclidean distances between the feature vector of the current frame, $\vec{v}(n)$, and that of the $n + j^{th}$ frame, $\vec{v}(n + j)$. Then the decision strategy is:

Loop for i from 1 to I_{max}
    until $| D_F(n, i) - D_B(n, -i) | > D_{min}$
    finally
    if $D_F(n, i) - D_B(n, -i) > 0$
        then associate frame n to its past
        else associate frame n to its future

Thus I_{max} constrains the observation range. Currently this value is set to 50 ms. The threshold D_{min} is a minimum distance threshold indicating when the difference between the two cumulative distance functions is significant enough to form an association. By terminating the search as soon as the threshold is exceeded, the algorithm self-adapts to capture short regions that are acoustically distinct. In addition, the algorithm assigns an association strength, $A(n)$, to each frame, which measures the maximum difference between D_F and D_B in the range of association. An example of the association waveform is shown in the top part of Figure 2. The positive-to-negative zero-crossings of the waveform correspond to potential acoustic boundaries. To minimize the effect of detecting small and insignificant acoustic changes, this association waveform is smoothed with a gaussian filter with sigma 0.005. For the example shown in Figure 2, this smoothed waveform is shown just below the association waveform.

The information in the smoothed association waveform can be captured in the form of a pulse train, also shown for the example in Figure 2. The pulse train provides information not only on the location of the acoustic boundaries, but also the boundary strength (by the height of the pulse) and abruptness (by the width of the pulse). In particular, we found that the height of the pulse is well correlated with the significance of the acoustic change. In other words, smaller pulses typically correspond to insignificant acoustic changes, or false boundaries. This observation is demonstrated in Figure 3, which compares the histogram of the pulse height for legitimate boundaries to that for false boundaries. Thus it is possible to set a boundary threshold and consider only those spikes whose height exceeds this threshold.

By varying both the boundary threshold on the pulse height, and the amount of smoothing performed on the association waveform, we can control the system's *sensitivity* to detecting acoustic boundaries. If the sensitivity is set too low, then the system may miss some of the legitimate boundaries. On the other hand, a high sensitiv-

**Figure 1:** A comparison of hair-cell versus DFT outputs.

26
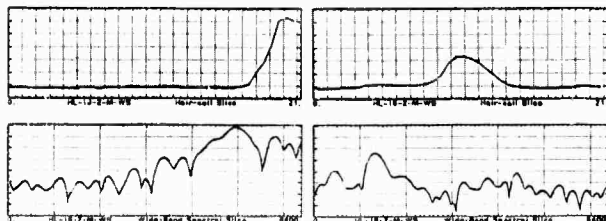
ity would tend to insert false boundaries. In our present implementation, the threshold and the amount of smoothing performed have been set to minimize the insertion *and* deletion error, based on a training sample of 200 sentences from 10 male and 10 female talkers. For the sentence shown in Figure 2, the acoustic boundary locations are superimposed as dotted vertical lines on the spectrogram. By comparing with the time-aligned phonetic transcription above the spectrogram, we see that most of the major acoustic boundaries have been located accurately.

Segment Classification

Once the boundaries have been determined, the regions within a set of boundaries are classified by a set of energy-related measures into one of four categories: sonorant-like (S), obstruent-like (O), silence-like (-), and murmur-like (M). To establish the general context of the region, the detection component next examines the energy contour and determines whether a given region signifies an energy peak, a valley, or a plateau. A murmur-like region is labeled prevocalic if it is followed by an energy peak, postvocalic if preceded by an energy peak, and medial if it represents an energy dip. Note that any isolated nasal, such as a syllabic nasal or a schwa nasal that has been labeled a single acoustic region will be labeled as S (about 5% of the detected nasals fall in this category). Currently, we make no attempt to recover these nasals since we are trying to label only murmurs adjacent to a vowel.

The broad classifier locates many nasal impostors because it utilizes a very straightforward context-independent algorithm with little speech knowledge. On both training and test data, the imposter to nasal ratio is approximately two to one. However, it is possible to rule out many impostors with several simple measurements before any detailed analysis of the murmur-like segments is made. This reduces the imposter to nasal ratio to approximately one to
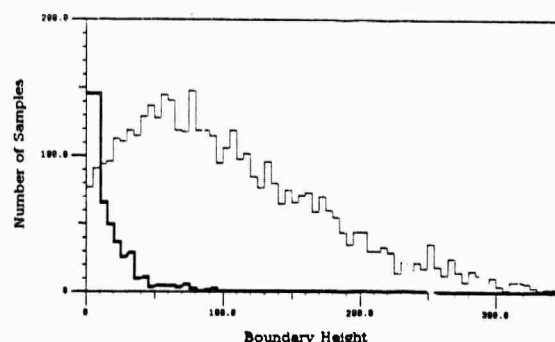
one, eliminating only a small fraction of the nasal candidates in the process.

Results

The nasal detection algorithm was evaluated on two separate databases. The first database consisted of 200 phonetically balanced sentences collected at MIT from 10 male and 10 female talkers. The recording was made in a sound-treated room with a lapel microphone. The second database of 165 phonetically balanced sentences was collected at C-MU from eight male and nine female talkers. The sentences were recorded in a computer room with a hand-held omni-directional microphone. All in all, there were 10 sentences each from 36 talkers and 5 additional sentences from 1 female talker in the C-MU corpus. None of the speakers or sentences were used for system training.

In order to evaluate the effectiveness of the nasal detection component, all labeled acoustic regions were mapped onto the time-aligned phonetic transcription. A phonetic label was associated with each region based on the phoneme that most overlapped the acoustic region. Using the criterion that a nasal has been detected if it overlaps a region by more than 50%, the system detects 70% and 71% of the nasals for the MIT and C-MU databases, respectively. For every potential nasal that the system detects, it also proposes 1.1 and 1.0 impostors for the two databases, respectively. A breakdown of the errors as a function of the phonetic context is shown in Table 1. We see that most of the detection errors occurred when the nasal was in a syllable-final position followed by an obstruent (as in "camp"), in a syllable-initial position preceded by an obstruent (as in "snap"), or between two vowels and realized as a nasal flap (as in "any").

Figure 2: The nasal detection component.

27

Table 1: Error Analysis for the Detection Algorithm

Phonetic Context	MIT Database (% of Error)	C-MU Database (% of Error)
V _ Obs.	72	66
Obs. _ V	14	17
V _ V	10	10

NASAL RECOGNITION

Strategy

The detection algorithm delineates regions in which a nasal murmur can potentially exist. However, some of these re...ns may indeed contain an impostor, i.e., a speech soun... is acoustically similar to a nasal murmur. The ...ge of our recognition system attempts to separate the nasals from impostors, which include front vowels, semivowels, voice bars, and weak voiced fricatives.

The classification system incorporates five robust sets of measurements on nasal murmurs, as suggested by an earlier acoustic study [2]. The five measures are:

- *Strength:* the average energy in very-low frequency band relative to the energy in low and medium-low frequency bands,

- *Stability:* the change in low-frequency and mid-frequency energy throughout the consonant,

- *Energy:* the difference in average energy between the consonant and the adjacent vowel, in both low and medium frequency bands,

- *Transition:* the maximum rate of energy change between the consonant and the adjacent vowel, in both low and medium frequency bands,

- *Change:* the amount of spectral change both within the consonant and between the consonant and the adjacent vowel.

Our earlier investigation revealed that the usefulness of the attributes depended on knowledge of the broad phonetic context. As a result, the broad phonetic context was used to divide the data into three categories: prevocalic, medial, and postvocalic. Further splits were also made in order to divide the imposter set into those which were more obstruent-like versus those which were more sonorant-like. This facilitated the subsequent decision-making process by allowing each classifier to select the features that best separated the nasals from the impostors for the given context.

Once the murmur-like segment had been sorted into a particular class based on its context and low resonance frequency, it was passed through a log likelihood classifier to produce a nasality score. The classifiers were trained on 400 sentences from an MIT database, containing 10 sentences from 20 male and 20 female speakers.

In order to avoid the difficulties of estimating multidimensional densities with a small number of tokens, and to avoid assuming some underlying distribution shape, the classifier assumed that the feature sets were statistically independent and estimated the density of each feature individually using a k-nearest neighbor approximation [1]. The nasal classification score, S, was produced by summing up individual log likelihoods for each feature.

$$S = \sum_{j=1}^{J} \log \frac{P_N(j)}{P_I(j)}$$

where $P_N(j)$ and $P_I(j)$ are the estimated probabilities of the jth feature for nasals and imposters respectively. J was typically around five for each classifier.

Results

As was the case for the evaluation of the nasal detection system, each acoustic region was assigned a phonetic label based on a 50% overlap criterion. Using this procedure, the classifier was able to correctly identify 80% and 79% of the murmur-like segments from the MIT and C-MU databases respectively. A breakdown of the performance is shown in Table 2.

To evaluate the performance of the entire system, taking into account both the detection and recognition parts, we compared the final output to the original time-aligned phonetic transcription. This evaluation produced two measures of the accuracy of the system, the *hit rate* and the *insertion rate*. The hit rate is defined as the percentage of time that an underlying nasal is detected and classified correctly (and overlaps with the time-aligned transcription). The insertion rate is defined as the number of impostors identified as nasal for each nasal in the database.

Evaluating the recognition system on the same 365 sentences produced hit rates of 54% and 59% and insertion rates of .12 and .18 per nasal for the MIT and C-MU databases respectively.

DISCUSSION

Nasal Detection

From Table 1, we see that the nasals missed by the detection algorithms consistently fall into several broad phonetic contexts. Nasal murmurs in these environments are typically articulated poorly. This is illustrated in Figure 4, which compares the duration of the nasals detected to that of the nasals missed. We see that the missed nasals are typically shorter than those detected. The missed nasals with durations of 50 ms or more either had a transcription error or had a subtle transition between the vowel and the nasal. Since our strategy is to locate only the robust nasals, we are not particularly concerned with these missed nasal consonants. As we can see from the spectrographic examples in Figure 5, vowels adjacent to poorly articulated nasals are usually heavily nasalized, suggesting that

Table 2: Error Analysis for the Classifier Algorithm

|	MIT Database Output (%)		C-MU Database Output (%)	
Input	Nasal	Imposter	Nasal	Imposter
Nasal	75	25	79	21
Imposter	15	85	20	80

acoustic information in adjacent vowels may carry the primary information on the presence of the nasal consonants.

Nasal Recognition

The majority of the recognition errors for the MIT database are due to confusions between nasals and semivowels, or between nasals and front vowels. In both cases, we suspect that information on formant frequencies and trajectories in adjacent vowels may be helpful. Over half of the errors in the C-MU database are due to confusions between nasals and front vowels. In this case we suspect the difference in the microphone may also play a role. Recall that the MIT data were collected using a lapel microphone, which tends to emphasize the low-frequency portion of the spectrum, whereas the C-MU data were collected with a far-field microphone. If microphone differences were indeed the source of this confusion, we may be able to improve system performance by including in the training set data collected from both microphones.

The present recognition algorithm utilizes only information in the nasal murmurs. In a previous study [3], we were able to determine vowel nasalization with some success. By incorporating acoustic information from both the nasal murmur and the adjacent vowel, we should be able to further improve system performance.

Figure 4: Histograms of the duration of detected and missed nasal murmurs.
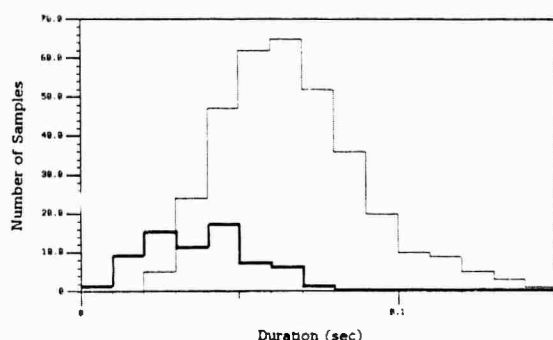
Figure 5: Spectrograms Illustrating the Missed Nasal Murmurs

Dance   Corner   Snap
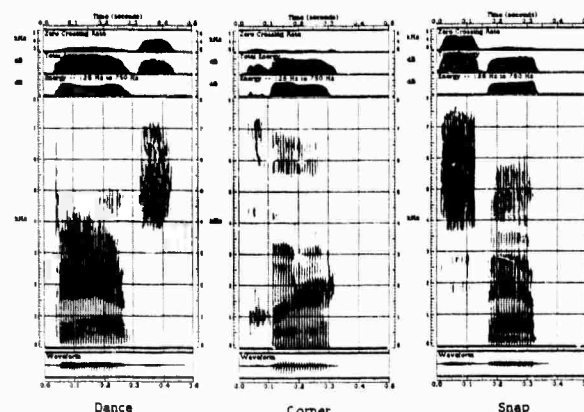
SUMMARY AND FUTURE PLANS

In summary, this paper presents a progress report on our effort to detect and recognize nasal consonants in continuous speech. Preliminary evaluations suggest that the algorithms are accomplishing what we expected, although further evaluation is definitely needed. Future work to improve system performance will include:

- refinement of acoustic features for recognition, including the use of formant frequency information, and

- the incorporation of information regarding the degree of nasalization in adjacent vowels.

REFERENCES

[1] Duda, R.O., and Hart, P.E., *Pattern Classification and Scene Analysis*, New York, John Wiley and Sons, 1973.

[2] Glass, J. R., "Nasal Consonants and Nasalized Vowels: An Acoustic Study and Recognition Experiment," S.M. Thesis, Massachusetts Institute of Technology, February 1985.

[3] Glass, J. R., and Zue. V. W., "Detection of Nasalized Vowels in American English," *Proc. ICASSP 85: IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1985.

[4] Rabiner, L. R., and Schafer, R. W., *Digital Processing of Speech Signals*, Englewood Cliffs, NJ: Prentice-Hall, 1978.

[5] Randolph, M. A., "The Application of a Hierarchical Classification Technique to Speech Analysis," paper presented at the 110th meeting of the Acoustical Society of America, Austin, Texas, 1985.

[6] Seneff, S., "A Computational Model for the Peripheral Auditory System: Application to Speech Recognition Research," paper to be presented at ICASSP-86, Tokyo, April 1986.

Pitch Tracking, Pitch Synchronous Spectra and Formant Tracking

by

William J. Majurski and James L. Hieronymus
National Bureau of Standards
Gaithersburg, Md. 20899

Abstract

A complete system for deriving formants from continuous speech is presented. The system contains a pitch tracker optimized for finding analysis frames which provide good spectra. The pitch tracker also provides a sonorant, obstruent decision and a voicing decision which is used by other components. The anharmonic pitch sychronous spectra are based on Hanning windowed regions of the pitch periods. Exactly which regions are analyzed is dependent on whether the region is an obstruent or a sonorant. Finally these spectra are used to drive a formant tracker based on the principle of maximum continuous length. The resulting system is being developed as part of the vowel recognition module at NBS.

Introduction

There have been a number of pitch trackers developed over the past 20 years. [1]-[7] Each one has been designed for a particular application. Some applications require smooth pitch frequencies, even though the real pitch of the first pitch period at the start of voicing is often low. The pitch tracker which is discussed here has been optimized for finding pitch periods which produce good spectra. In glottalized regions, it will find pitch periods of approximately the length of regular pitch periods in that part of the sentence. In the course of developing this pitch tracker, many very strange looking waveforms were examined. In each case the pitch period boundaries are selected which produce the best spectra. We built our own pitch tracker because our sonorant identification modules account for coarticulation by looking at formant transitions between the sonorant and adjoining obstruents. The pitch in the transition region is difficult to track accurately, but is of great importance for our pitch synchronous analysis of these regions.

Pitch Tracker

The goal of the pitch tracker is to produce high quality spectra suitable as input to a formant tracker based on peak finding. The algorithm used is fast, at an early stage performing data reduction on the input speech and using only reduced data thereafter. Detecting the true pitch produced by the talker is not a goal of this project. Only those measures with increase the performance of the formant tracker are used. Particular emphasis is given the the edges of voiced regions where transitional information may be present in the resulting formants. As byproducts of pitch detection, a measure of sonority and an estimate of syllable boundries are produced.

A pitch period is the amount of time between subsequent excitations of the vocal tract during voicing. The exponential waveform decay during the pitch period is a commonly recognized feature of a voiced region of speech. The algorithm employed by this pitch tracker [7] utilizes this exponential decay of amplitude to detect the presence of voicing and the alignment of the pitch period.

This algorithm uses a parameterized representation of the input speech. The speech waveform is low-pass filtered so as to include approximately the lowest two formants. Using the filtered waveform, zero crossing locations and peak heights between zero crossings are computed. All calculations are based on zero crossings and peak heights with one exception. The transition between voiced and unvoiced regions is detected using a sum of squares energy calculation on the original waveform.

This algorithm detects the pattern of decaying amplitudes which appear as decending steps on a plot of peak heights.[Figure 1b] Each flight of steps represents a pitch period. Within a pitch period the largest peak is either the pitch pulse or the glottal opening, the opposite signed peak preceding the pitch pulse. Using apriori knowledge of the orientation of the pitch pulse the pitch peak is labeled. This algorithm alone correctly labels pitch periods within open vowels. Pitch labels in other voiced regions typically require one or more forms of correction. Pulses which do not represent pitch pulses can be labeled, valid pitch pulses can go unlabeled and occasionally the pulse following the pitch pulse will be labeled instead. In order to clean up the mislabelings, an overall estimate of pitch is made by finding the largest region of constant pitch. Restrictions on feasible pitch range are used to restrict the choices.

The most frequent labeling error is marking non-pitch pulses as pitch pulses. This is corrected by identifying and deleting the errant label. Pitch periods containing an extra labeled pulse within it are frequently adjacent to periods with correct pitch. This is the easiest case to detect and correct. [Figure 2] In other places an entire voiced region will be labeled with double pitch marks and thus not supplying a easy starting point for correction. Lacking an appropriate neighbor, canidate pitch pulse amplitudes are investigated. A correlation of canidate pitch peak amplitude is used to choose peaks for deletion. [Figure 1c] When every second canidate is significantly lower in amplitude than its neighbor the higher amplitude pulses are labels as pitch pulses and the lower amplitude pulses are deleted.

If the amplitude difference between neighboring peaks is not significant (roughly 30 percent) then the extra peaks are labeled as pitch pulses. This produces an abnormality in the pitch frequency but has been found not to significantly degrade the resulting synchronous spectra.
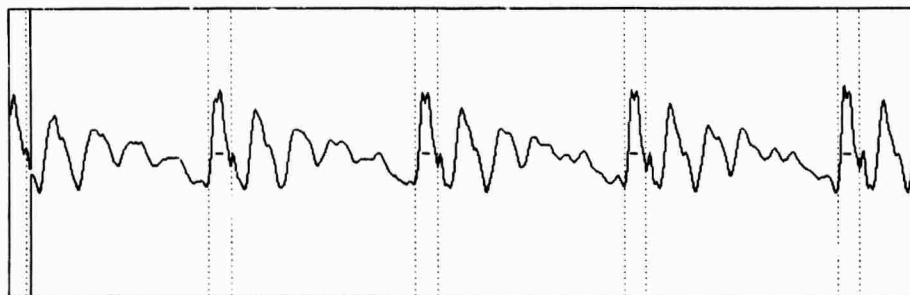
Figure 1a. Original Waveform with pitch pulses labeled.
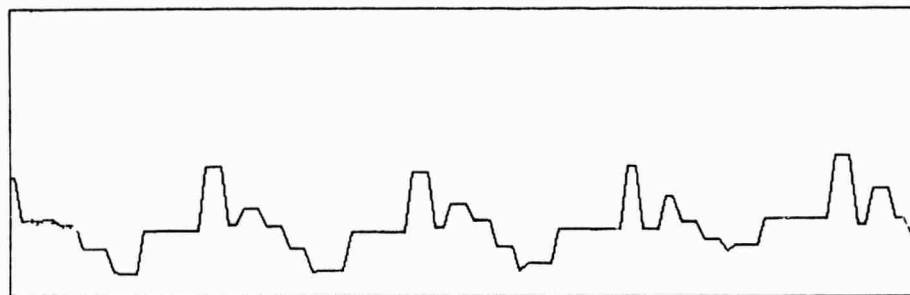
Figure 1b. Peak heights - Time aligned with original waveform
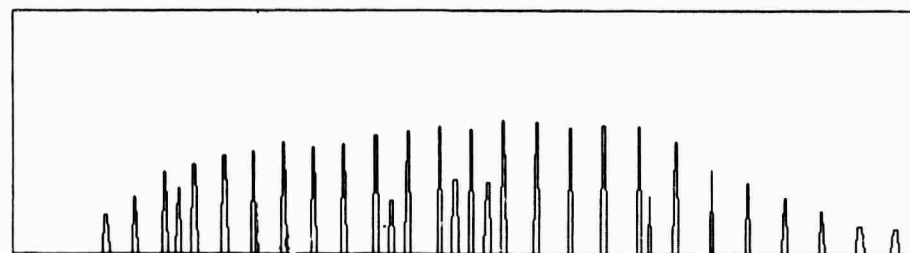
Figure 1c. Pitch Pulse magnitudes. Incorrectly labeled pulses are identified by their magnitudes.

Other areas require interpolation or extrapolation of pitch marks. Voiced obstruents typically produce a waveform which shows little amplitude decay and as few as two peaks, one full cycle, per pitch period. These do not trigger the amplitude decay based detector and must be detected seperatly. [Figure 3] If the voiced obstruent has adjacent to it a more open voiced region containing well formed pitch periods then pitch labeling is extrapolated into the voiced obstruent. As before, zero crossings from the filtered waveform are used to align the pitch marks. The pitch labeling is extended until it joins another already labeled region or an energy threshold detector indicates the end of voicing.

Formant Tracker

The pitch synchronous spectra are derived from anharmonic analysis. Pitch synchronous spectral analysis of various types has been developed over the past 25 years. Anharmonic pitch synchronous analysis was first discussed by Hess [15]. In the simplist case, the waveform is windowed with a Hanning window which begins at the zero crossing before the principal excursion or excitation pulse and ends at the zero crossing before the next principal excursion. Then the windowed waveform is padded with zeros to form a window long enough to give the frequency resolution desired in the FFT.

31
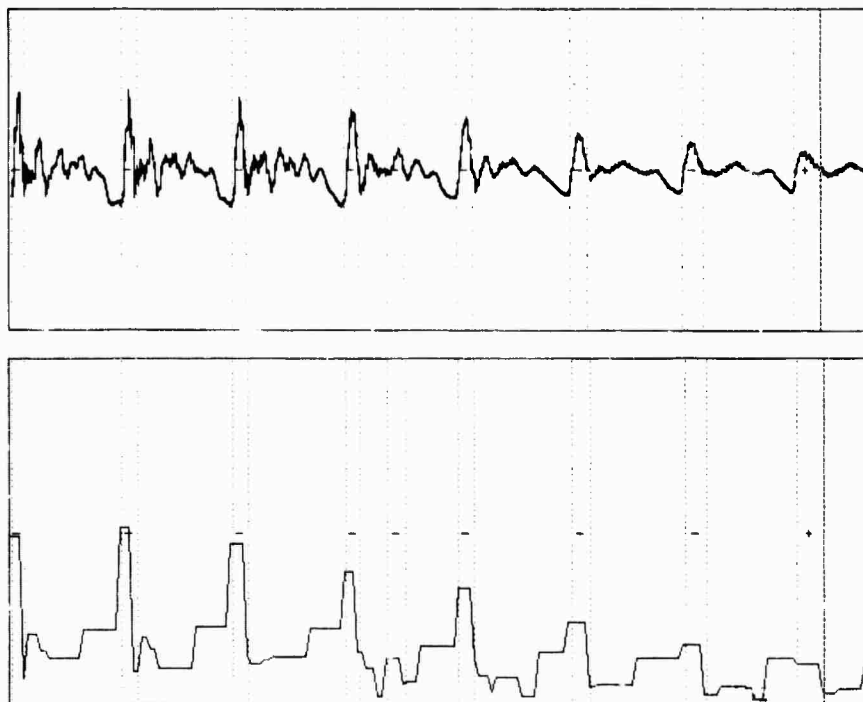
Figure 2.   Original waveform above and peak heights below showing
mislabeled pitch pulse.
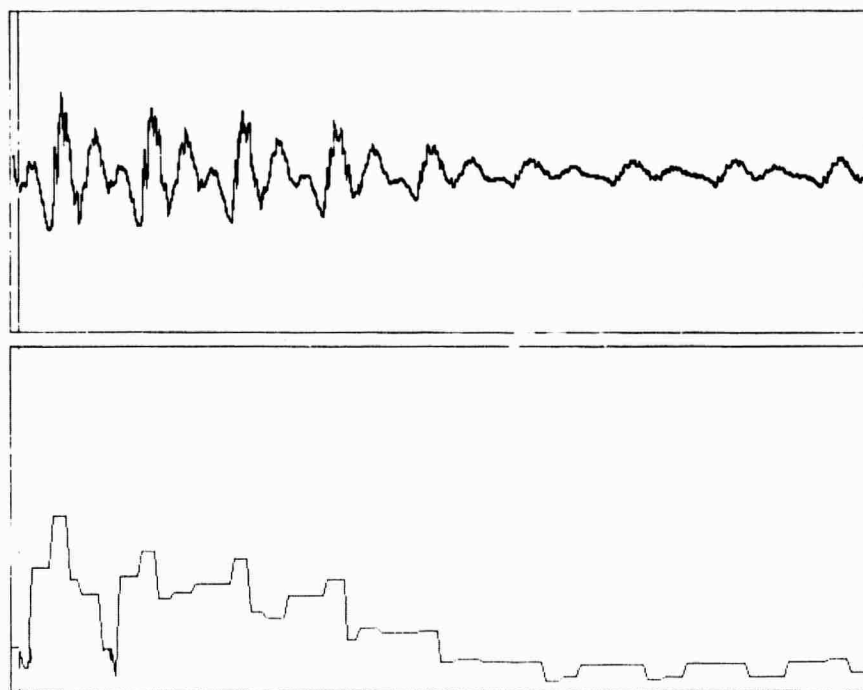
Figure 3.   Original waveform above and peak heights below showing
transition into a closure causing ramp pattern to
deteriorate.
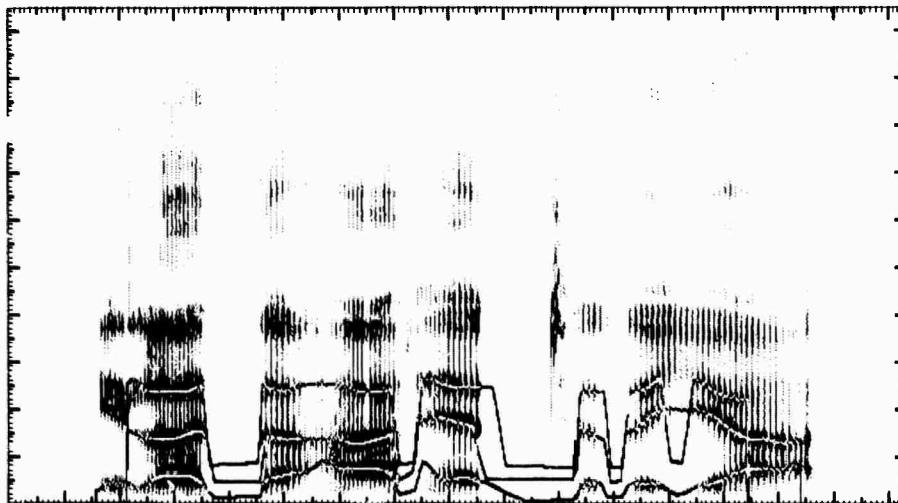
Figure 4.   Spectrogram showing calculated formant tracks.

In our implementation, the spectral analysis of the sonorants and obstruents are done differently. The Hanning window for the sonorants is approximately 75roughly to the glottis closed portion of the waveform. For obstruents there is very little energy in the waveform, other than the principal excursion. Therefore the window is moved forward so that the principal excursion in in the middle of the Hanning window. This differentiation between the processing for sonorants and obstruents leads to superior spectra and thus makes the task of formant tracking easier.

The formant tracker is based on the idea of mimicing the way a human spectrogram reader finds the formants. Humans seem to look for more or less continuous dark regions in the spectrograms, and label these formants. Many formant finding algorithms have been developed in the past based on as many models of what a formant is. These formant trackers had varying degrees of success. Our algorithm works by finding the main peaks in the spectrum in a sonorant region and then extending these peaks into less sonorant regions. After a structure which represents the length of the regions with continuous spectral peak trails has been built, then these trails are examined and assigned to the formants. Currently up to six peak trails are found in each sonorant region.

The formant tracker uses synchronous spectra calculated using pitch labels computed by the pitch tracker. Regions of contiguous pitch periods are labeled voiced regions with all other time being accounted for as unvoiced regions. Formant tracking is restricted to voiced regions.

Formant tracking proceeds in three stages: peak picking, peak tracking and formant tracking. Peak picking finds the major peaks in the synchronous spectra. Peak tracking finds ridges in the spectra within a voiced region which correspond to formants. Formant tracking takes the per voiced region peak tracks and combines them and assigns formant number labels (ie. F1, F3 ...) to each formant.

In the first stage, peaks representing vocal tract resonances are extracted from each synchronous frame in each voiced region. Peaks are chosen in amplitude decending order to insure the major resonances are extracted. Even with synchronous spectra, closely spaced peaks whose spectral skirts overlap significantly are difficult to discriminate. As each peak is identified, a parabola is fit to the peak using a least squares calculation. The parabola is subtracted from the spectra. This leaves any overlapping peaks intact for subsequent extraction.

In peak tracking, spectral peaks within a voiced region are linked to form peak traces. Peak tracking is performed in a way which is analogous to what we do when looking at a spectrogram. Peaks are linked to peaks of similar frequency in adjacent frames. Since this must operate on glottalized regions, peaks are matched with adjacent frames and with frames two away. The peak lists are then pruned to exclude nonesential tracks.

In the formant tracking stage, each voiced region is divided into thirds. Peak tracks which are present in the third closest to an unvoiced region are considered to border the unvoiced region. Unvoiced regions will be refered to as breaks. Canidate pairs of peak tracks for matching are found. To be considered, a track must boarder the break and be the best match in frequency. To qualify, both border tracks must choose the other as its best match and the next closest track must be greater than double or less than half in frequency. Less stringent rules are then applied to complete the matching. Finally, a consistancy check is made on the matches. This section looks for formant tracks which change formant affiliation thus indicating a bad match was made between peak tracks. Shifts in match assignments are made to correct the inconsistancies. We have not yet considered cavity affiliation switches in this algorithm.

Work on the pitch tracker and formant tracker is not yet complete. This paper is intended as a progress report of ongoing work.

33

References

[1] Ben Gold, "Computer program for pitch extraction,"
    J. Acoust. Soc. Am. **34** (July 1962) 916–921.
[2] D. R. Reddy, "Pitch period determination of speech
    sounds," *Comm. ACM* **10** (June 1967) 343–348.
[3] M. M. Sondhi, "New methods of pitch extraction,"
    IEEE Trans. Audio Electroacoutics, **AU-16** (June 1968)
    262–266.
[4] C. M. Harris and M. R, Weiss, "Pitch extraction by
    computer procession of high-resolution Fourier analysis
    data," *J. Acoust. Soc. Am.* **35**,(1963) 339–345.
[5] M. R. Schroeder, "Period histogram and product spec-
    trum: new methods for fundamental-freqency measure-
    ment," *J. Acoust. Soc. Am.* **43** (Apr. 1968) 829–834.
[6] A. M. Noll, "Cepstrum pitch determination" *J. Acoust.
    Soc. Am.* **41** (Feb. 1967) 293–309.
[7] Neil J. Miller, "Pitch Detection by Data Reduction,"
    IEEE ASSP.(1975)

Compensating for Vowel Coarticulation: a Progress Report

by

James L. Hieronymus and William J. Majurski
National Bureau of Standards
Gaithersburg, Md. 20899

Abstract

Seven English monothong vowels were studied in continuous sentences. The purpose of the study was to determine what methods are likely to be successful in compensating for coarticulation in all vowel and consonantal contexts. A method by Kuwabara has been examined in detail. The Kuwabara compensation improves the separation of the vowel regions in a space composed of the first and second formant in Japanese. Important issues are where to measure the formant "target" frequencies, how to obtain good formant tracks, measuring speaking rate accurately, and how to label vowels accurately.

Introduction

Vowel articulation, that is the propensity of nearby phonemes to alter the characteristic frequencies of vowels, is a problem in continuous speech recognition. The effect of a nearby liquid or glide (i.e., /l/, /r/, /y/, or /w/) is very large. Some back vowels are actually effected so much by a consonant that they become "fronted." Using a conventional classifier on these vowels results in a misclassification. Also continuous speech tends to centralize the formant frequencies of all vowels, especially those with secondary stress.

Coarticulation has been studied by several groups in the last 15 years using isolated CVC or VCV words and these words embedded in carrier phrases. Stevens and House [2] studied the perturbation of vowel formant target frequencies by consonantal context. They studied yy American English vowels. They concluded that vowels in symmetrical CVC environments are centralized relative to their formant target frequencies in isolation or in the hVd context.

Lindblom [3] studied vowel reduction in Swedish. He concluded that the effect of coarticulation could be modeled by a decaying exponential. The constant multiplier and decay time constant were dependent on the vowel and consonant. The longer the vowel (more stress), more closely the normal vowel targets were reached. This study was done on CVC words (where the beginning C and the following C were the same consonant) embedded in carrier phrases. The data was provided by one talker.

Ohde and Sharf [4] studied the effects of voiced stops on vowel formant targets. They found that the effect of voiced stops on vowel reduction was greater for preceding consonants than for following consonants. Once again this work was done on isolated CVCV utterances.

Broad and Fertig [5] studied vowel coarticulation for the vowel /I/ in CVC monosyllables consisting of all possible combinations of 23 consonants plus a silence element.

Analysis of variance showed that the superposition principle characterized the results very well. The transition functions for initial and final consonants were in general asymmetric.

A major difference between CVC words in isolation or in carrier phrases and continuous speech is that the timing of the articulators can be more carefully planned in the CVC case. This means that symmetric CVC's are more nearly symmetric, and the vowels in CVC's are longer than in connected speech. Consonant clusters in continuous speech can have a cumulative effect on the vowel formants. These complex interactions must be studied in continuous speech, using the results from the CVC studies as a guide. Also prepausal lengthening and duration changes due to stress mean that the vowel durations are much more variable than the isolated CVC word case.

Several methods of compensating for vowel coarticulation have been proposed. However, none of these methods has been tested on large amounts of speech data. The most promising methods use some measure of the formant slopes near the vowel to compute an adjusted formant frequency.

A recent paper by Kuwabara [1] tested coarticulation renormalization for vowels in Japanese sentences. Since Japanese is a CV language, these results may not hold for a language with a more complicated syllable structure like English. This method used a symmetric Gaussian function in time to compute a renormalized formant trajectory for the first and second formants. The technique was motivated by perceptual studies of vowel sequences and auditory modeling. Using the 5 vowels /i/, /e/, /a/, /o/, and /u/ from Japanese, the method was successful in eliminating most of the confusion of these vowels in F1-F2 space. Before the method was applied there was considerable confusion due to centralization.

A similar scheme was used by us to compensate for vowel coarticulation in American English. Seven American English monothong vowels (/i/, /I/, /eh/, /ae/, /a/, /o/, and /u/) were studied. Two continuous speech data bases were used which contained 1853 exemplars of these vowels. The data shown in this paper is for 225 vowels out of the 1853. The formant tracks obtained automatically were examined for accuracy. Approximately 50 of these vowels were not used in the study because of formant tracking errors due to nasalization. The major questions to be answered by this study are 1) Does this method work well for English? 2) Does it compensate adequately for coarticulation from semi-vowels? 3) Can the use of other functions than Gaussians improve the performance of the compensation? 4) Do adjustments of the width of the Gaussian have to be made to account for coarticulation at different speaking rates?

Some other compensations are also being tested. One uses decaying exponentials to compensate for coarticulation. Another one uses different curves derived from the work of Broad [6] for compensation. Broad proposed several families of decaying exponentials for each class of consonants and the principle of superposition for the effects of preceding context and following context. None of the data from these other compensations is shown in this paper.

The compensations have the potential to improve the vowel recognition in continuous speech.

Speech Data Bases

Two data bases of continuously spoken sentences were used in the experiment. The first, the CMU Coca-Cola Database, was collected and labeled at Carnegie-Mellon University by the Speech Recognition Group. It contains 600 sentences from the Harvard list, 10 spoken by each of 32 male and 28 female speakers. The speech was directly digitized using a 16 bit AD converter sampling at 16 k samples/second. A cardioid microphone was used in a computer terminal room, which provided fan noise and some low level background speech.

The second speech data base was collected and labeled at NBS. It contains 20 sentences spoken by 5 male and 3 female speakers. The sentences contain words with the seven vowels /i/, /I/, /eh/, /ae/, /a/, /o/, and /u/ in most contexts from the neutral h_d context to glides and semivowels. The speech was collected in a sound isolation booth using a Shure SM-10 close talking microphone on one channel and a B&K pressure microphone on the other channel of a Sony PCM-F1 digital audio processor. The output of the close talking microphone was direct digitized using a 16 bit AD converter sampling at 16 k samples/second. The talkers produced the sentences at their normal speaking rate and at a "fast" rate. The fast speech resulted from instructing the talkers to speak as rapidly as possible without obvious mispronunciations. A list of the sentences is given in Table 1.

1. He led the hot head to the yacht.
2. The rat rode the door to the weedy lot.
3. The rude lad had hoped to wed the doll.
4. Yes, the yak wore a red yoke.
5. He wooed her with his lewd wit.
6. She woke hearing a roar in her ear.
7. The yeast dough rose against the lid.
8. You'd better load the real gun.
9. We'll lead them to the lead rod.
10. You're getting rid of this wad.
11. The deer wagged its tail and ran to the well.
12. The crude fiber deal yielded no profit.
13. The doll liked the yacht tour very much.
14. The well rod rode against the rule.
15. The old yoke decorated the room.
16. The wad of gum was on a lid behind the door.
17. Will we dare hold the rally next year?
18. That year the Dallas gang war was in a tar yard.
19. The rule was yelled by the leering judge.
20. According to lore Dirty Larry wallowed in the rill.

Table 1. Sentences in the NBS Coarticulation Data Base

The number of exemplars of each vowel in the two data bases is shown in Table 2.

The CMU and the NBS data bases had only a small number of /uw/ vowels, so we were not able to study these as extensively as the other vowels.

Distribution of Vowels			
Vowel	Sex	CMU	NBS
/i/	m	208	98
	f	193	88
/I/	m	226	80
	f	204	84
/eh/	m	173	81
	f	163	88
/ae/	m	134	42
	f	122	41
/a/	m	148	63
	f	149	64
/o/*	m	30	0
	f	23	0
/ow/	m	68	53
	f	65	51
/u/	m	34	25
	f	28	27

* (non-dipthongized /ow/)

Table 2. Number of exemplars of each vowel

Formant Tracking

Automatic formant tracking is a difficult task which requires care at every step of the procedure. The formant tracker which we have developed uses anharmonic pitch synchronous Fourier transforms as the input data. Major peaks in the log magnitude squared spectrum are selected using amplitude and area. Data structures which retain the time history of the peak frequencies are searched for continuous formants. Only as a last step in the process are the formants assigned labels as first formant, second formant, etc. Since the data used in this paper only require first and second formants, only this data is shown here. A detailed discussion of this formant tracker and its performance will be presented in a later paper. Figure 2 shows the output of the formant tracker superimposed on a wide band speech spectrogram.

0.0000   Wide-Band Spectrogram   0.9739
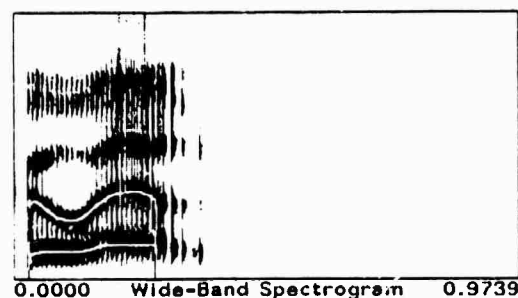
Figure 2. Formant Tracker Output

At present strongly nasalized vowels with split first formants can have the nasal formant identified as the second formant. A special technique is being developed to eliminate this error.

36

Vowel Formant Targets

In order to find the vowel target frequencies, two techniques were used. The first replicates the usual technique of taking the temporal midpoint of the region with the vowel label. Figure 3 shows the male data with 225 vowel targets using this technique.
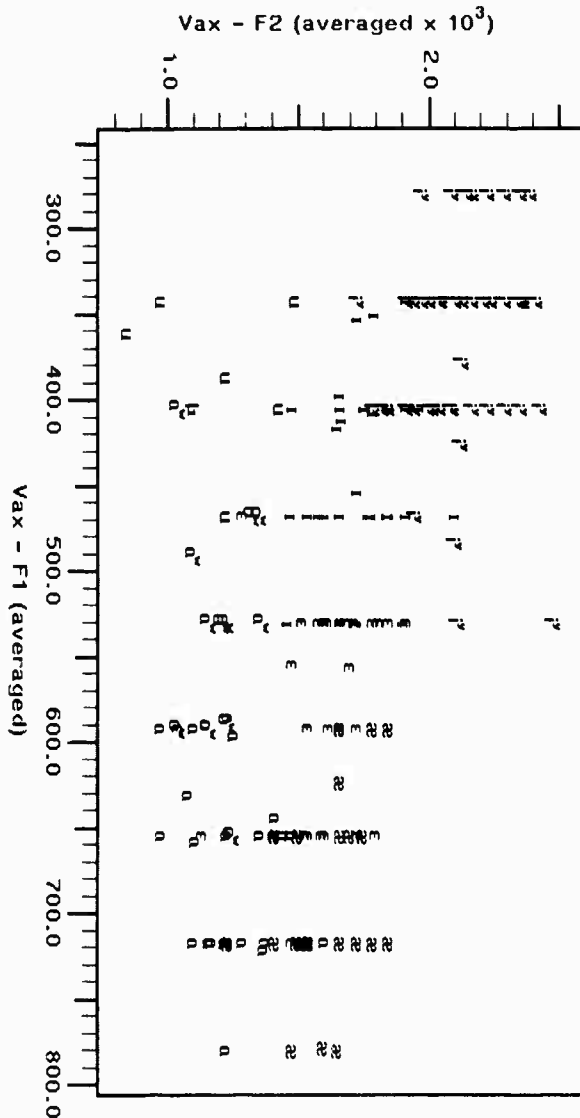
Figure 3. Male Vowel Target Frequencies Method 1

Figure 4. Male Vowel Target Frequencies Method 2

The second technique is to take the target to be the place with the minimum average slope for the first and second formant. Thus if the "steady state" region is shifted due to coarticulation, it will be captured in this technique. Figure 4 shows the result of using this technique on the vowels presented in Figure 3.

The first technique seems to produce better results in that the vowels of the same identity cluster in more compact regions. However the effect is not so pronounced to clearly indicate that the first technique is best. The vertical rows of labels in the figures are due to the quantization of the FFT, which has a resolution of 60 Hz. The results for the females is similar but will not be shown because of the size limit for this paper.

Renormalized Vowel Formant Targets

The Kuwabara method was applied to the vowel formant trajectories and the resulting renormalized vowel formants obtained. Only method 1 for determining the vowel target frequencies is shown, since method 1 seems to give superior results. One detail of our present implementation differs from the original Kuwabara method, we do not extrapolate the formants into the consonants for voiceless consonants. This should effect a maximum of 1/3 of the data. Instead we do a linear extension of the formants into the neighboring unvoiced regions. This discrepancy will be corrected in future work. Figure 5 shows the male vowels with targets obtained from method 1.

Discussion

By examining Figures 3-5 in detail we see that the Kuwabara method does not produce the same spectacular results for American English that it does for Japanese. Figures 6 – 9 show the Kuwabara results from Japanese for comparison. Figure 6 and 7 are for vowel – vowel transitions and Figure 8 and 9 are for vowel – consonant transitions.
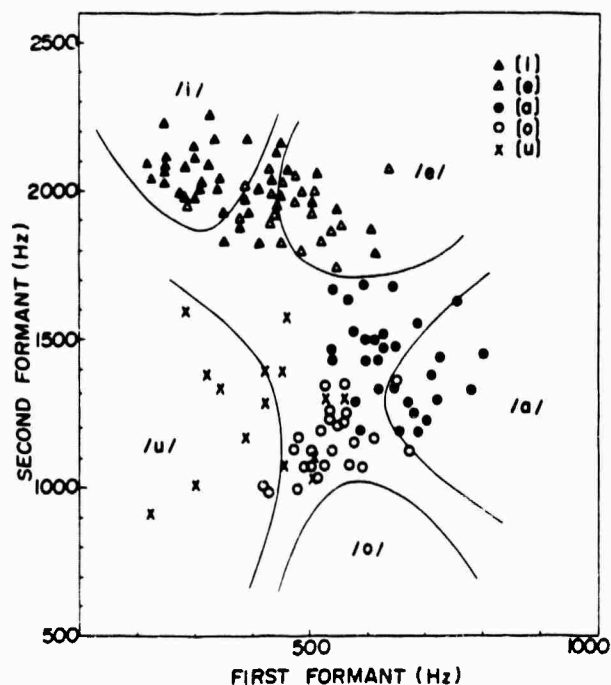
Figure 6. Target Frequencies for 5 Japanese Vowels

Figure 5. Renormalized Target Frequencies Method 1
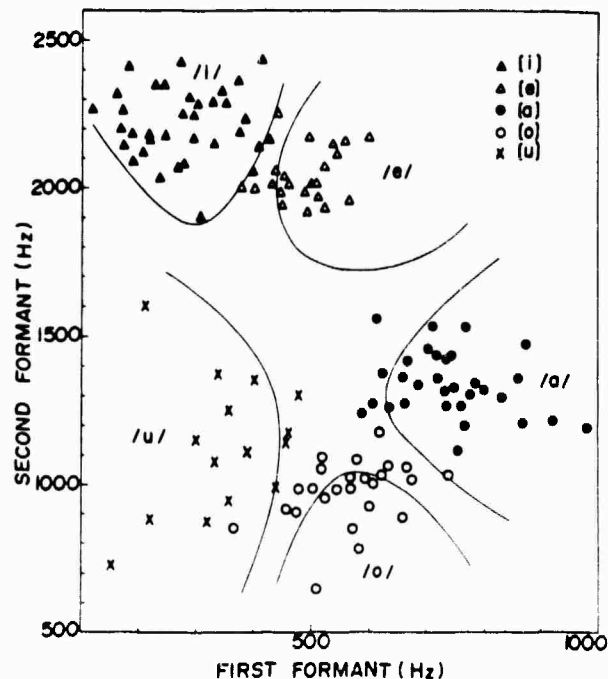
Figure 7. Renormalized Japanese Target Frequencies

38
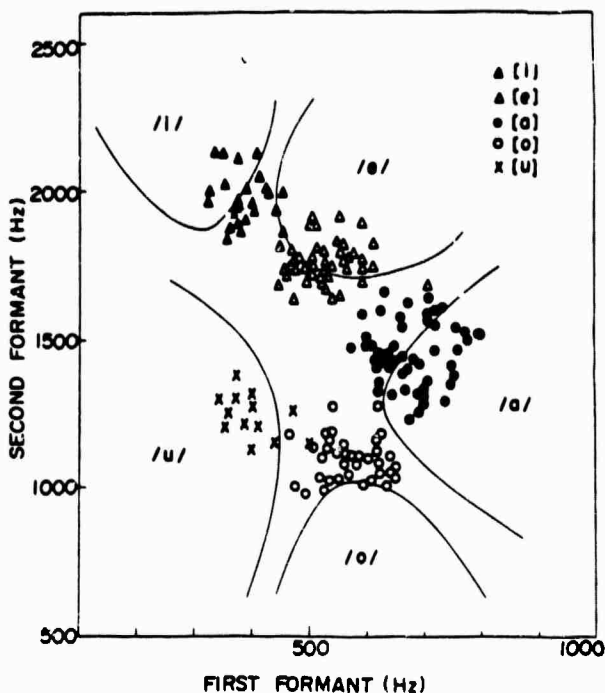
Figure 8. Target Frequencies for 5 Japanese Vowels
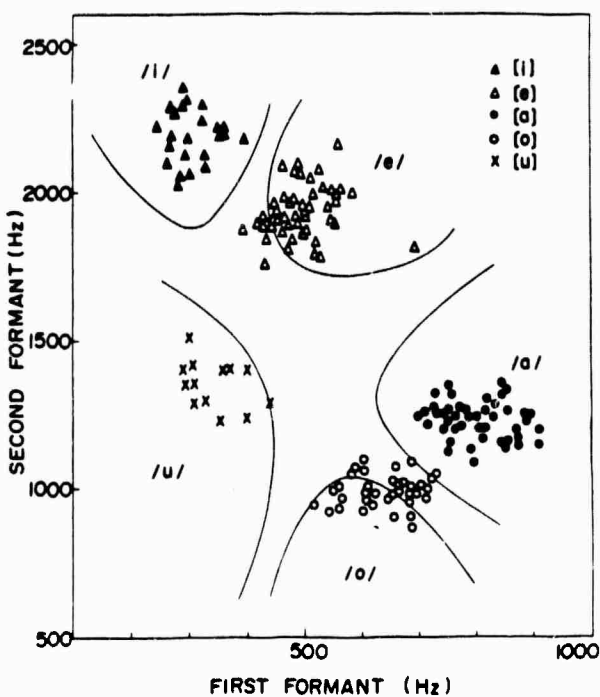
Figure 9. Renormalized Japanese Target Frequencies

Perhaps this difference in performance is due to the CVCV structure of the Japanese language. The Kuwabara renormalization shifts some of the target frequencies in the direction of the usual regions for these vowels. The /I/ vowels which are mixed with the /iy/ vowels are moved to regions of higher F1 and thus generally out of the /iy/ region. However the separation between the regions is not large. On the other hand one of the /ow/ vowels was moved into the /uw/ region. The separation between /iy/ and /uw/ were made less by the renormalization. The vowel /iy/ was shifted to a second formant frequency above 3000 Hz. in many cases. Thus, this method does not consistently shift the vowel target into its canonical position in F1-F2 space.

It is possible that the Kuwabara renormalization will be useful in improving the first choice performance of a vowel classifier, because on the average it throws the vowel formants in the "right direction." To see this more clearly Figure 10 shows a scatter plot of the difference between the Kuwabara formant and the normal formant frequency for the first and second formant.
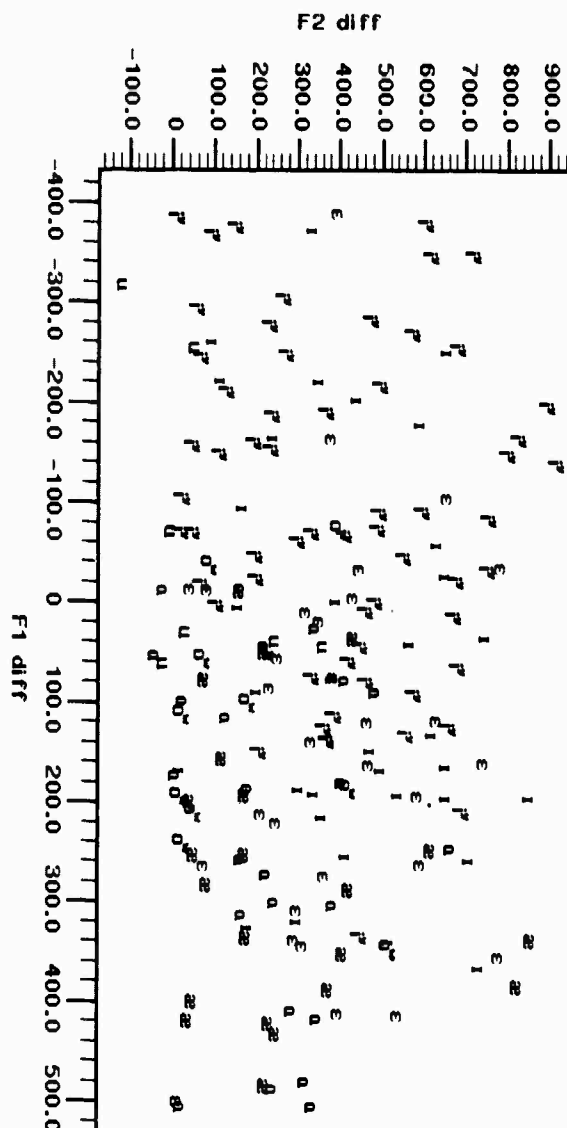
Figure 10. English Vowel Corrections

39

From the plot we see that the Kuwabara renormalization gives generally symmetric correction to the formants. The corrected values are not large. Figure 11 – 12 shows a blow up of the most central region of the F1 – F2 vowel space for the ordinary formants and the renormalized ones.
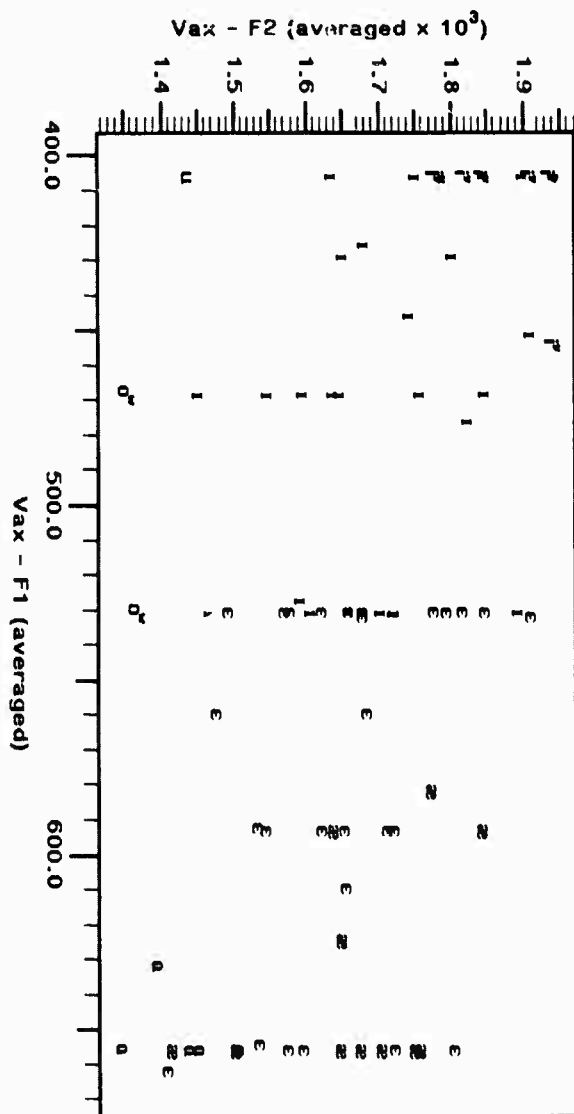
Figure 11. Target Frequencies for English Vowels

We will examine the possibility of adding the Kuwabara renormalization to a CMU style classifier in the next two months.

Acknowledgements

Discussions with Victor Zue and Ron Cole were very helpful in the formative stages of this project. Thanks to the CMU Speech Group for the labeled "Coca-Cola" Data Base and a pitch tracking program. Thanks to the MIT Speech Research Group for the Spire and Cart Speech Analysis Tools. Mac Townsend and Ken Spagnola provided programming support at NBS. This work was funded in part by DARPA Contract N0003984PD41304.
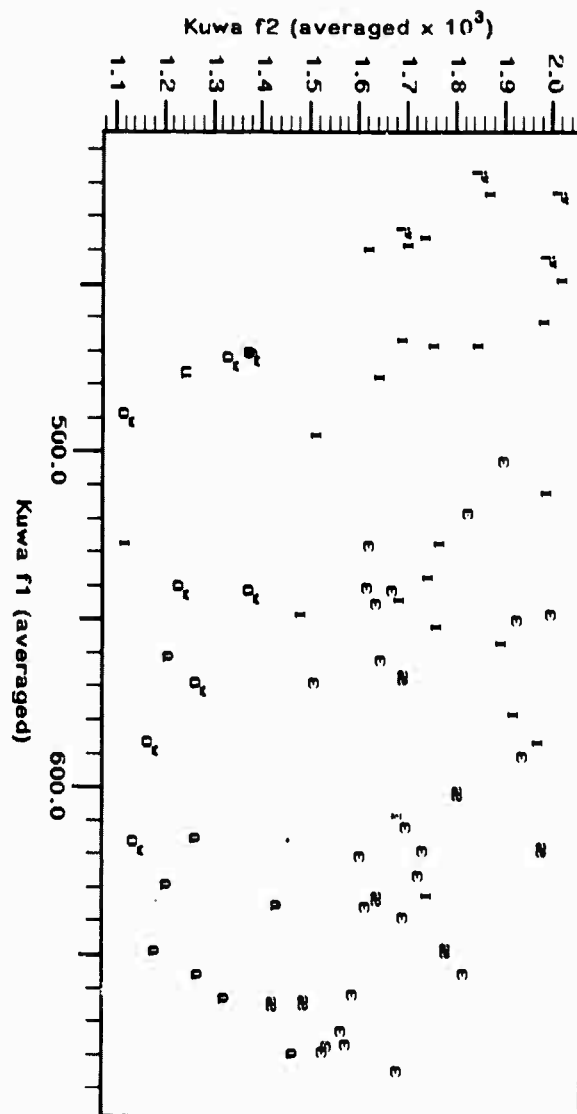
Figure 12. Corrected Vowel Targets

References

[1] H. Kuwabara, "An approach to normalization of articulation effects for vowels in connected speech." *J. Acoust. Soc. Am.* **77** (Feb. 1985) 686–694.

[2] K. Stevens and A. House, "Perturbations of Vowel Articulations by Consonantal Context: Acoustical Study," *J. Speech Hear. Res.* **6** (1966) 111–128.

[3] B. Lindblom, "Spectrographic Study of Vowel Reduction," *J. Acoust. Soc. Am.* **35** (Nov. 1963) 1773–1781.

[4] R. Ohde and D. Sharf, "Coarticulatory effects of voiced stops on the reduction of acoustic vowel targets," *J. Acoust. Soc. Am.* **58** (Oct. 1975) 923–927.

[5] D. Broad and R. Fertig, "Formant-Frequency Trajectories in Selected CVC-Syllable Nuclei," *J. Acoust. Soc. Am.* **47** (Oct. 1969) 1572–1582.

[6] D. Broad, "Vowels in Context: Dynamics, Statistics, and Recognition," *Proceedings of the 1983 Santa Barbara Conference: Towards Robustness in Speech Recognition,* Speech Science Publications, Apple Valley, Minn.

PHONOLOGICAL STUDIES FOR SPEECH RECOGNITION

Jared Bernstein, Gay Baldwin, Mike Cohen,
Hy Murveit and Mitch Weintraub

Speech Research Program, SRI International
Menlo Park, California 94025

1. INTRODUCTION

This report presents some results of the first nine months work at SRI International on the problem of how to accommodate dialectal and phonological variation in American English pronunciation in the design of large vocabulary speaker-independent speech recognition systems. The particular system component that we are focussed on is Lexical Access (or Word Hypothesizing), although differences in pronunciation between and within speakers need to be taken into account in the design of many components of a recognition system.

Lexical access involves associating a string (or lattice) of input symbols with sequences of lexical items. Designing a recognition system component that accomplishes this association can be seen in two parts: empirical and algorithmic. Empirical studies are needed to determine, for example, the circumstances in which /sr/ palatalizes or in which schwa devoices or deletes. Algorithms need to be designed to support the empirical studies and to use the empirical knowledge within a system. SRI's research plan has been to pursue the facts first and work on implementations and system integration second. Our empirical results should be generally useful, and are not necessarily tied to any particular system approach.

This report covers three tasks of our first year's work: a database study; a rule-tool system; and t-rules across three speakers.

2. DATABASE STUDIES

Our concern was with phonological and phonetic differences between spontaneous speech and the speech produced when people read from prepared text. A prime motivation for studying these differences is the use of read speech to train speech recognition systems. Systems are being designed to adapt to a new speaker on the basis of an brief enrollment passage that is read. In principle, a read passage should be adequate to delineate many aspects of a person's speech, including oral and nasal resonant frequencies, prosodic parameters, dialect features such as vowel inventories and qualities, and phonological style. Presumably, the physical properties of the vocal tract will be about the same in reading as in spontaneous speech. However it may be that linguistic aspects of speech are systematically different in the two modes.

For collecting a database of speech samples on which to train and test speech recognition systems, it is more efficient to collect specific, prepared materials, rather that the very large samples of spontaneous speech needed to encounter specific phonological phenomena. If one is recording read speech, what elicitation procedure will yield speech that is most like spontaneous? Examples like the one shown in Figure 1 very easy to generate. An educated woman produced the word <competitive> with two flaps in spontaneous speech, yet when asked to read the same words, she does not flap the second /t/ even

when reading quite fast, and she consistently flaps the first /t/ even when reading **very** carefully. This behavior suggests two things: **(1)** for some phenomena and some speakers, reading may not yield the same forms found in normal speech, and **(2)** the probability that an underlying segment will be realized in a certain form (in a particular environment) may be much more stable for one segment-in-environment than for another over a range of speaking modes.

COMPETITIVE
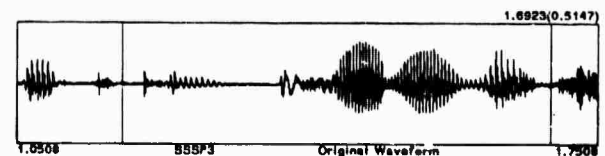
Spontaneous   [flap] [flap]
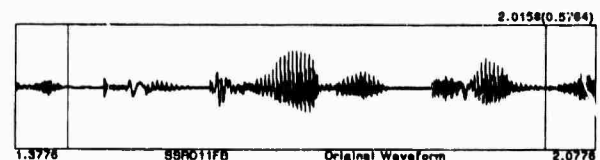
Fast Reading   [flap] [t]

Normal Reading   [flap] [t]

Careful Reading   [flap] [t]

Figure 1.

We set out to measure speaking rate and deletion of phonetic segments in read vs. spontaneous speech and to use these measures to find out which reading mode (if any) is closest in form to spontaneous speaking. This paper reports data from three speakers who were employees at SRI. These three are one fast talker (M1), one precise talker (F) and one more or less neutral talker (M2). Each speaker was recorded in a interview-like exchange for 30 minutes. The conversation was conducted in a quiet room with the interlocutors seated about 6 feet apart; the subject was wearing a head-mounted, noise cancelling microphone. The conversation was then transcribed by a secretary and searched for items of interest like frequently used words that contain consonant clusters. The most frequently used clustery words in written English include "problem, probably, question".

About 30 sentences were selected from the transcriptions for each speaker. These sentences included words of interest and according to the secretary's transcription they were well formed grammatically and would not embarrass the speaker when asked to read them. Of these 30 sentences, about 20 per speaker turned out to be fluent in the spontaneous recording. These 20 sentences were then typed on cards and the each speaker was asked to read the 20 sentences from their own conversation. Following this reading, they were asked to read the same sentences "very fast", and last they were asked to read the sentences in a slow and careful manner as if they were "s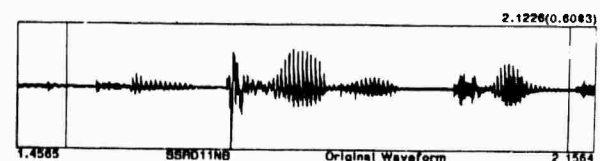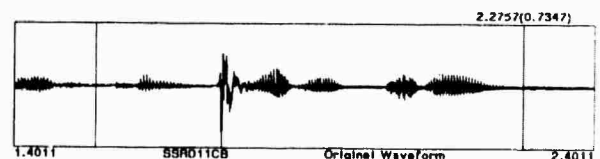peaking to a hard-of-hearing person over a poor telephone connection". These three instructions define normal, fast and slow reading. Of the 20 sentences, only about 15 per speaker were replicated word for word in all four conditions (spont., fast, normal, and slow). Typically, in one or more of the read versions of an excluded sentence, the speaker had changed, reversed or left out a word. Thus the materials reported here are 180 sentences (3 speakers x 15 sentences x 4 versions).

Figure 2.

All the durations and rates cited here are for actual speaking time; pauses greater than .25 seconds in length were subtracted out. Since sentence prosody mainly affects durations in the region from the last stressed syllable to the end of the sentence (Klatt 1976,1979; Umeda 1976), the measurements excluded the tail-end of each sentence from the onset of the last stressed vowel. The sentence displayed in Figure 2, "Probably after the baby is born, we'll go back", was measured from the burst of [p] in "probably" to the onset of the [a] in back, with the "Against the callousness" would be measured from the [g] burst in "against" to the [a] onset in "callousness".
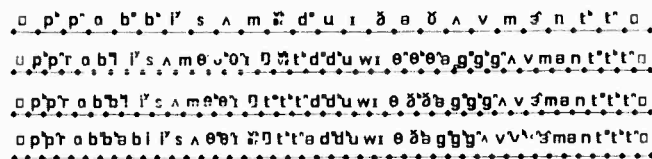
Figure 3.

A precise phonetic/acoustic transcription of the four versions of the 15 sentences was made for each of the sentences. An example "probably something to do with the government" is displayed in Figure 3. The top transcription is of the spontaneous version. The 2nd, 3rd and 4th are fast, normal and slow reading respectively. The top (spontaneous) version is missing any sequence of segmental events corresponding to "-thing to", although when the utterance is heard, it is very clear that this is what the speaker said. Perhaps the durations of the [m] and the [d] subsume the rhythmic place of "-thing to"; however there is no glottal stop in the [m] that would yield a clear percept of "sump'm". In most cases sonorant nuclei are counted as one segment, since it is very hard to decide on the presence or absence of pre- and post-vocalic sonorants next to an obstruent. For instance, it is very hard to differentiate [od] from [old]. Note also in Figure 3 that the three read versions of the sentence have a full stop and burst transcribed for the /g/ in "government", but the spontaneous version has a velar approximant 'gamma'. This does not count as a deletion; the three microsegments of the /g/ count as one realization of the /g/, as does the voice velar approximant, 'gamma'.
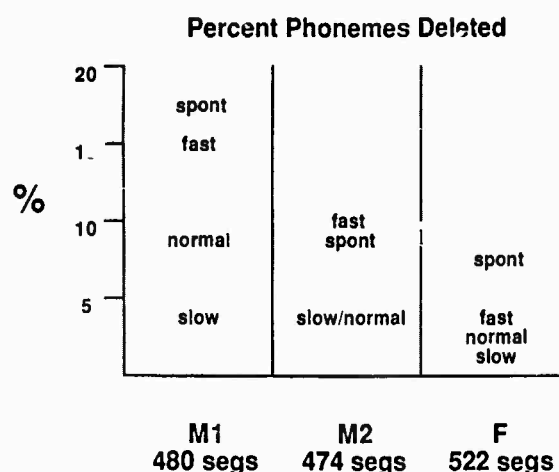
Figure 4.

Figure 4 shows the percentage of segments that were deleted in the various versions of the material. The fast male speaker (M1) deleted 18% of the segments in spontaneous speech, 15% in fast reading, 9% in normal reading, and 4% in slow, careful reading. The precise female speaker (F) showed the same ordering of the speaking conditions, but the magnitudes are less. The 'neutral-ish' male (M2) deleted about the same number of segments in fast and spontaneous, and a much smaller number in normal and slow reading. For two of the three speakers, no reading condition was phonologically similar to their spontaneous speech by the deletion measure. But perhaps deletion is a function of speaking rate, and the spontaneous speech is faster.

The data displayed on Figure 5 show that for speaker M1 (and the others as will be seen in the next two slides), the spontaneous speech is like the fast reading in absolute duration, but more like the normal reading when measured in phonetic segments per second. Thus, assuming that deletions are simply a function of speaking rate won't work.

In Figure 5, the values displayed are medians and quartiles. The leftmost mark represents the values found if you measure the slow reading rate of each sentence and subtract from it the rate (in segments/sec) of the normal reading of that sentence. Thus, for sub-

Speech Rate
(segments/second - normal reading rate)
Subject M1
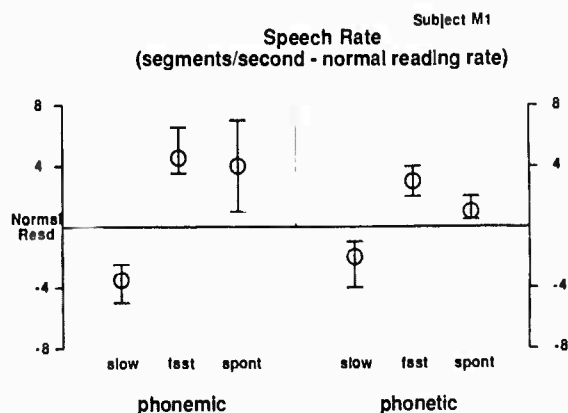
Figure 5.

ject M1, the slow readings tend to be about 3.5 segments/second slower than normal reading, when counting **PHONEMIC** segments. The phonemes, the linguistic elements, in each version of the sentence are the same, so the segments per time in the left side of Figure 5 are just a scaling of time. On the right side of the slide, the values shown are differences in **PHONETIC** segments per second. Here, if one version of a sentence is completed in a much shorter time than another version, the speech rate in segments per second may not be any different, if some number of segments in that version of the sentence were skipped (or 'deleted'). Thus the phonemic rate approximates linguistic material per time, and the phonetic rate approximates the articulation rate per time.

The normal reading rate for the first male speaker, M1, averaged about 16 phonemic segments per second, with the fast and slow readings about 4 segmentsper second faster and slower, respectively. In elapsed time or phonemic rate, his spontaneous productions were most like his fast reading. However, referring back to Figure 5, since his average rate of segment skipping was quite different in the four conditions, the situation looks different when you measure phonetic segments per second. Although fast, normal and slow reading keep their relative positions, the rate differences are less when counting phonetic segments. More importantly, in phonetic segments per time, the spontaneous speaking rate is most like the normal reading rate. That is, in elapsed time or linguistic material per time, the fast reading is most like spontaneous speech, but in articulation timing, normal reading is most like the spontaneous material.

Speech Rate
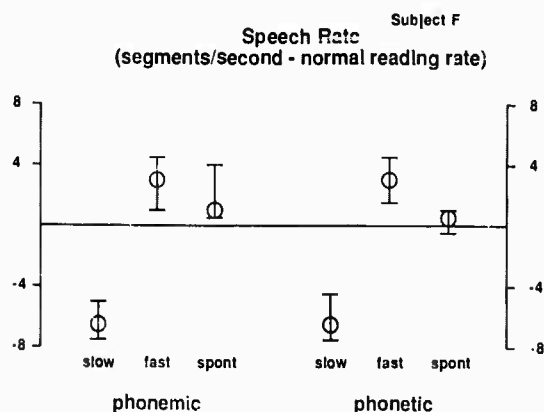**(segments/second - normal reading rate)**
Subject F

Figure 6.

The female speaker, F, shows a similar pattern: In elapsed time and

in percent deletions, fast reading is most like spontaneous speech; but in the rate of production of articulated sounds, the normal reading is most like her spontaneous speech.

Speech Rate
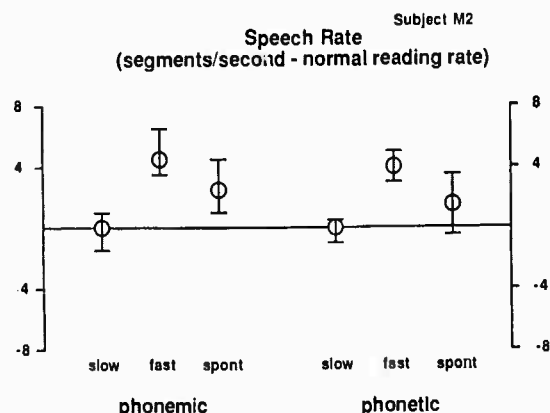**(segments/second - normal reading rate)**
Subject M2

Figure 7.

The second male speaker shows no differences between slow and normal reading, and other differences are somewhat attenuated. Nothing is very clear cut in Figure 7, but the data are not particularly inconsistant with the pattern of the other two speakers.

People cover more linguistic material per time in spontaneous speaking than in normal reading. However, when instructed to read faster, they mostly increase rate by speeding up each segment that is spoken. This is in contrast to spontaneous speech, where fast rate is accomplished more by skipping segments.

With regard to selecting a procedure for recording read materials to train speech recognizers, normal reading may be the best for studies of phonetic durations and coarticulatory phenomena, but fast reading will most likely yield a better approximation of the phonological patterns of the speaker. No reading seems to yield both.

Relevant to the design of lexical access with a speech recognition system, we need to know where and in what circumstances people are likely to delete segments. None of the reading material provides an accurate or complete picture of this aspect of people's speaking habits.

3. NETWORKS AND RULE TESTING

SRI has constructed a facility for the study of phonological variation. The tools include:

(1) an interactive facility to write phonological rules and apply them to baseforms, change rules and test their effects on the phonological representation, transcribe speech using phonological rules, and test a set of rules on a database of transcribed speech.

and (2) a graphical facility to display and manipulate network representations of possible phonetic realizations.

These tools can be used with a database of transcribed sentences to iteratively update and test the adequacy of a set of phonological rules. This facility separates the linguistic knowledge that goes into a set of phonological rules, and the actual programs that use these phonological rules. In this way, the linguistic knowledge of a system is explicitly represented. We describe both of these aspects of the system: the **form** of the phonological rules and networks, their **implementation**.

Networks consisting of nodes and arcs are used to represent the

pronunciation utterances. Phonological alternations can be represented by the application of rules to a network. Such networks and rules can serve several purposes, including as a framework for letter to sound conversion, the enumeration of possible pronunciations, fitting duration models, and for advanced data-base searching.

For instance, rules can be used to transcribe speech if there is a dictionary of baseforms for words and a set of rules to transform this baseform into a network of possible pronunciations. Using the interactive network facility, one can select those labels that were in the actual recorded pronunciation. When the labeling is complete, a menu pops up to provide a file name for storing this transcription. Integrated with an automatic alignment program, this process can be made semi-automatic.

If the actual pronunciation is not allowed in the pronunciation network, the user can pop into the rule editor, write the appropriate rule to generate this pronunciation and then restart the above procedure. In this way, a person can transcribe speech and at the same time develop the set of rules that can account for the observed speech data.

Figure 9

3.1. Form

3.1.1. Pronunciation Network Representation

The network representation used by SRI consists of a data structure composed of nodes and arcs. Each symbol in the network is contained as the label on an arc in the network. An arc's label can be used to represent words or phonemes. If the arc's label is NIL, then it is treated as a null arc. The purpose of a null arc is to allow this arc to be skipped. A series of arcs (with phonetic labels) is displayed in figure 8.
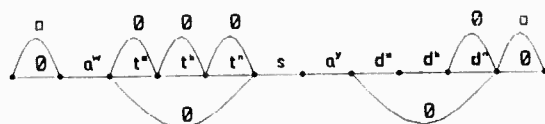

Figure 8.

The above network represents the word <outside>.

The data structures that are used to represent the network, nodes, and arcs, are NETWORK, NODE and ARC. The NETWORK not only has lists of nodes and arcs, but various pointers into SPIRE data structures. Each NODE belongs to a NETWORK and keeps track of its own ARCs in and out. The ARC structure itself embodies the most interesting new features of these networks. Beside the usual 'from and 'to and 'label, each ARC has slots to identify the arc and rules from which it was generated in the current network, a time alignment with a SPIRE file, and pointers to corresponding ARCs in higher or lower level NETWORKs that represent the same material.

3.1.2. Rule Editor

In order to facilitate the writing of rules, a "rule editor" has been constructed. A picture of this facility can be seen in Figure 9. It consists of a regular zmacs editor, along with several stationary menu's. By clicking on any of the menu's, corresponding "rule text" will appear in the zmacs window at the location of the cursor. By moving the cursor around and clicking on the appropriate menu's, one can conveniently write rules.
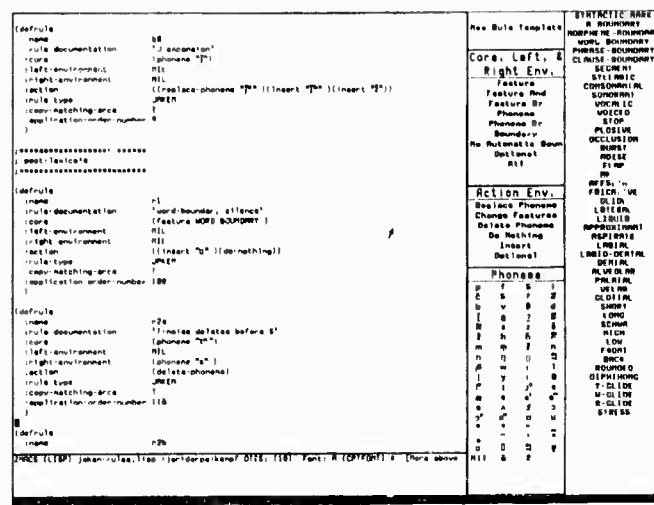
The rules are saved to a file just like any other lisp file. If one wishes to modify a rule, one simply changes the text of the rule, and then recompiles the rule. Then, if this rule set is applied to a network, all earlier versions of this rule are automatically deleted.

3.1.3. Rule Syntax

When one uses the rule editor, the following default display is provided to the rule writer:

```
(defrule
        :name
        :rule-documentation
        :core
        :left-environment
        :right-environment
        :action
        :rule-type               SRI-1
        :copy-matching-arcs      T
        :application-order-number 0
)
```

The contents of each rule is organized in four slots which are the: "left environment", "core", "right environment", and "action". Each of these four slots can contain a series of clauses. Each clause consists of a test that is applied to an arc, to determine if that arc satisfies that clause. There exist a predefined series of clauses that are available to the rule writer. The following predefined clauses are can be used ONLY in the left, core, or right environments:

1. (feature foo) e.g. (feature voiced) -- This tests the arc's feature structure to determine if the arc is voiced. If it is desired to test that the arc is not voiced, one would write: (feature (voiced nil)) Only one feature may be present in this clause.

2. (feature-and foo foobar) e.g. (feature-and voiced (stress 2)) -- This tests the arc's feature structure to determine if all the features listed are satisfied. The above clause would test an arc to determine if it was voiced and had a stress level of 2. All the features are either T or Nil, except for stress, which is 0, 1, or 2.

3. (feature-or foo foobar) e.g. (feature-or voiced (sonorant nil)) -- This tests the arc's feature structure to determine if any of the features listed are satisfied by this arc.

44

4. (phoneme "string") e.g. (phoneme "p") -- This test the arc's label to determine if it is the same as the phoneme in the clause. Distinctions are made between upper and lower case.

5. (phoneme-or "s1" "s2" "s3") e.g. (phoneme-or "p" "t" "k") -- This test the arc's label to determine if it any of the following phonemes.

6. (boundary foo-boundary) e.g. (boundary morpheme-boundary) -- This tests the arc's feature structure to determine if it is a morpheme boundary. This is very similar to the "feature" clause. The difference between this and the feature clause is in the way that the clause is compiled. The rule compiler (discussed in a later section) automatically inserts the clause (optional (feature morpheme-boundary)) everywhere, except when it sees a boundary clause.

7. (no-automatic-boundary) -- This is an instruction to the rule compiler for it not to insert the clause (optional (feature morpheme-boundary)) between this clause's two neighboring clauses. It does not translate into a clause itself.

8. (optional (clause)) e.g. (optional (feature voiced)) -- The optional clause may be followed by any test, except for clauses 6 and 7 (this is because 6 and 7 are instructions to the rule compiler). In addition to containing any predefined clause type, this may contain any lisp code (see example below).

9. In addition to these predefined clauses, the user can combine the first five clauses in any lisp function that he desires. Predefined clauses "boundary" and "no-automatic-boundary" cannot be contained in lisp expressions. For example:

(optional (or (feature-and nasal dental) (phoneme "th")))

For each clause in the core environment, there must be a corresponding clause in the action environment (except for the "no-automatic-boundary" clause in the core, since it does not generate any code). For each optional clause in the core, there must be a corresponding optional clause in the action environment. Insertions have no corresponding clause in the core. Each corresponding clause determines the action that is to be performed on the arcs that match the core clauses. If the arcs are copied, the action is taken on the copied arcs. If the arcs are not copied, the action is taken on the matching arcs. The following predefined clauses are can be used ONLY in the action environment:

```
(replace-phoneme "foo")
(change-features foo foobar ...)
(delete-phoneme)
(do-nothing)
(insert "foo")
(optional (delete-phoneme))
```

Additional predefined clause types can be written. Some possible new clause types might include (Part-of-speech-p foo) to test if 'foo is the grammatical class of the word dominating the current node, or (tri-morph-frequency-greater constant) to test if the arc is part of a three morph sequence with frequency of occurence greater than 'constant. Another more immediately useful addition would be (replace-phoneme-list "foo" "foo1" "foo2").

The following sample rule demonstrates the syntax:

```
(defrule
  :name                      B7
  :rule-documentation        "expands unstresses initial-syl IY to IH"
  :core                      (and (phoneme "IY")(feature (stress nil)))
  :left-environment          ((feature word-boundary)
                              (optional (feature-and segment (syllabic nil))))
  :right-environment         (feature (syllabic nil))
  :action                    (replace-phoneme "IH")
  :rule-type                 CMU
  :copy-matching-arcs        T
  :application-order-number  0
)
```

Note that for the core, right, and action environments there is only one clause and it appears by itself. When more than one clause is present (as is shown in the left environment), they must be enclosed in a list.

3.1.4. Interactive Interface for Examining Networks

In order to facilitate the examination of networks, a "network examiner" has been constructed. This facility can be seen in Figure 9. It consists of a graphics pane in which the networks are displayed, a notification pane which prints out instructions, a command menu pane which provides menu commands to manipulate the networks and their corresponding displays and a lisp pane for lisp expressions.

3.2. Implementation

3.2.1. Rule Compiler

The rule compiler takes the information contained in the rule and compiles it in a form that is used by the rule application program. Each clause in the rule is compiled into a "rule-clause" lisp structure. One of the slots of this clause is called "match-compiled-function". The rule clause that the person writes will typically contain a macro such as (phoneme "p"). The function "phoneme" is a macro that expands out into code that tests the label of the arc that is currently under consideration. This code is compiled using the following lisp code:

(compile (gensym) '(lambda (arc) ,clause))

This creates a random function (which is bound to the variable returned by gensym), which contains the compiled code that is to be applied to an arc. Therefore, any clause that the person writes can assume that it will be applied with the variable "arc" bound to the arc that we are currently testing. When the rule clause is applied to an arc, the following lisp code is used:

(funcall (match-noncompiled-function clause) arc)

If this returns T, then the arc successfully satisfies the rule clause's requirement. If this returns nil, then the arc did not satisfy the clause requirement.

3.2.2. Rule Application Strategy

3.2.2.1. Rule Ordering

For each rule set, the rules in that set are ordered by the rule number specified in the rule. However, taking a fixed set of rules, there are several different strategies by which they can be applied to a network.

```
1. Loop for each rule in the list-of-ordered-rules
     Loop for each arc in the network
          Test if the rule applies to the network starting
               at this arc.
          If rule is successfully applied,
               then modify the network.

2. Loop for each arc in the network
     Loop for each rule in the list-of-ordered-rules
          Test if the rule applies to the network
               starting at this arc.
          If rule is successfully applied,
               then modify the network.
```

45

This is a choice of algorithms if the rule set is to only be applied once to the network. The current rule application program uses algorithm number 1. Algorithm number 2 is also available if desired. The rule application program keeps count of the number of rule that have been applied in a variable called *rule-applied-count*. The actual algorithm that is used to apply the rules is listed below

```
1. Loop selecting the first rule remaining in the List-of-ordered-rules
        2. Collect all the rules that have the same application
                order number as this rule into Temp-rule-list
        3. Set *rule-applied-count* to 0
        4. Loop for each rule in Temp-rule-list
                5. Loop for each arc in the network
                        6. Test if the rule applies to the network
                                starting at this arc.
                        7. If rule is successfully applied, then modify the
                                network, and increment *rule-applied-count*.
        8. If *rule-applied-count* > 0, Go to step 3
        9. Delete all the rules of temp-rule-list from the list of
                remaining rules and iterate.
```

3.2.2.2. Matching a Rule to a Series of Arcs in a Network

When a rule is applied to an arc in a network, this arc is tested against the first clause in the rule. If this arc successfully matches the first clause, then we want to proceed onto testing the next clause. However, if there are several arcs which follow this first successful arc, a recursive algorithm is used, where the next clause is applied to each of the following arcs. If any of these arcs successfully satisfy the clause, then the system recurses with the next clause test on the following arcs. A simplified version of the algorithm is described "conceptually" below:

```
(defun test-rule (remaining-clause-list last-positive-arc)
  (cond ((null remaining-clause-list)
          ;; there are no clauses remaing to be matched
          (possibly-apply-rule-to-network rule))
        (t
          (loop for arc in (following-arcs last-positive-arc)
              with clause = (first remaining-clause-list)
              do
                (cond ((funcall (match-compiled-function clause) arc)
                        ;; the arc successfully satisfies the clause
                        (store-arc-as-matched arc clause)
                        (test-rule (cdr remaining-clause-list) arc)))))))
```

The actual algorithm used is somewhat more complicated to take care of optional clauses, Null arcs and the possibility of applying rules across word boundaries where each network is one word long.

When there are no more clauses left, the rule is potentially applicable to the network. Since the rules are iteratively applied to the network, a test must be performed to determine if this rule has already applied to this same series of arcs in the network. If it already has applied to the same series of arcs, then the rule is not applicable to be applied again. Otherwise, the rule is applied to the matching arc list.
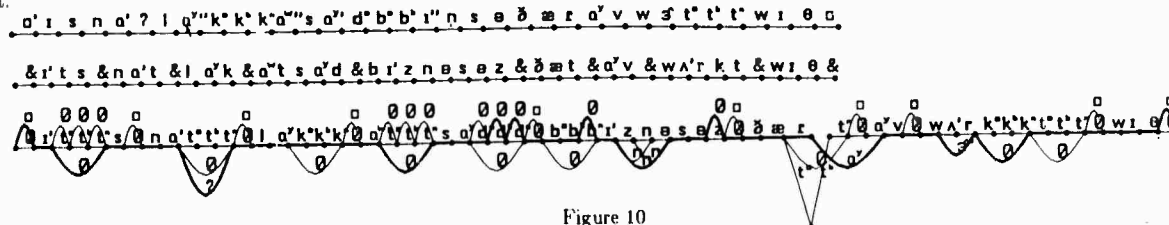
3.2.2.3. Applying a Rule to a Matching Series of Arcs

When a rule is applied to a matching series of arcs, if the arcs are to be copied, then all the arcs matching left, core, and right clauses are copied. Then, for all the arcs that match core clauses, the corresponding "action" is taken on each of these arcs, as specified in the action clause and the appropriate bookeeping is performed.

3.2.3. Testing Rules

Given a series of rules and a database of transcribed speech, one can test that the rules can account for all the observed pronunciations in the database. This is done by a process which is similar to the process of transcribing speech described in a previous section. The process of testing a rule set involves (1) producing the baseform pronunciation from the dictionary, and then applying the relevant rules to the baseform sequence to produce a fuller network that represents all the pronunciations consistent with the baseforms and rules, (2) use the interactive network facility to find the longest path through the network of possible pronunciation that matches the actual pronunciation network. This path is highlighted on both networks. If this path does not reach the end of the network, then the user knows that additional rules are needed to account for the observed pronunciation. The user can then go into the rule editor and write the appropriate rule, recompile and reapply the rules.

An example of rule testing is shown in Figure 10. The upper network is a hand transcription of the spontaneous utterance "It's not like outside businesses that I've worked with." The second network is the string of baseforms automatically retrieved for that sentence, and the third network is shows all the possible phonetic sequences that can be derived from the application of the phonological rules for this speaker to the second, 'baseform', network. The path through the third network that corresponds to the hand transcription has been automatically identified and highlighted.

4. /T/ ACROSS SPEAKERS

We are studying the phonology of /t/ across three speakers in year one so that we might develop a model of across- and within-speaker phonological variation for a limited environment. With this model and the model derived from our complete phonology of one speaker, we hope to estimate the size and complexity of a model of general phonological variation.

4.1. Issues

The basic issues involved in this study are:

- What forms are realized for /t/s?

- What environments are correlated with these forms (phonemic environment, speech rate, word or word frequency, speaker, speech style)?

- What types of functions can be derived that estimate the probability of an observed form given the environment of a /t/?

- How can these probabilistic estimates be improved by observations of other near-by forms observed in the speech?

Figure 10

46

Basic issues needed to be resolved in order to complete the above are:

- How do we define a form in an objective manner? For instance, what differentiates a flap from a d-stop?
- How are speech rate, speech style, and word frequency defined?

4.2. Materials

For an initial study of /t/ across speakers, the spontaneous speech forms realized by three carefully chosen speakers are being studied.

The speakers used were among 20 speakers chosen from 150 speakers in an earlier study (Bernstein, Kahn and Poza; 1985). The 20 speakers were chosen to represent the range of speaker characteristic and speech styles evident in the 150 speaker sample, and the three speakers used here were one fast male speaker (JK), one neutral male speaker (SH) and one precise female speaker (SS). JK is subject "M1" in the database study reported in Section 2; the other two speakers are different from subjects "M2" and "F". These speakers were interviewed in half hour sessions. Spontaneous speech was chosen because the phonological processes in effect in spontaneous speech differ significantly from that of read speech (see section 2, above), and should be similar to preferred manner of speaking commands to speech recognition systems.

4.3. Tools

We have developed several tools to aid us in observing the /t/-forms and to help gather statistics about their occurrence. The first is a digital tape recorder based on the Symbolics 3600 LISP Machine. This allows us to keep a session of recorded speech (one of the interviews) on line. The speech is recorded direct to disk and played back from it. Recording requires a Digital Sound Corp. A/D converter attached to the LISP Machine, but playback does not. This tool aids in quickly locating and transferring portions of the spontaneous speech to another tool for classification and data collection.

The orthographic record of the recorded sessions was manually typed into text files. This text was converted to an approximate phonemic representation by the use of the PROSE-2000 text to speech converter.

Given the text and the phonemic transcriptions, the speech can then be searched for phonemic environments under study. A facility to compute a nominal speech rate for that context will also be developed. The speech rate will be defined as the phrase duration predicted by Klatt's (1979) model phonetic durations, divided by the observed duration. Thus, if a speaker produces a sequence of words in half the time that Klatt's rules would predict, the nominal rate would be 2.0.

All the above tools are integrated into a data collection facility. The goal of this tool is to collect data of discrete variation of /t/, making the categorical decisions necessary in an objective manner. A secondary goal is to facilitate the quick collection of such data, as a large amount of speech must be analyzed in order to compute meaningful statistics. Also, time offsets in the disk recording files are stored so that particular data points can be recovered and reanalyzed quickly.

The are several important issues relevant to the design of such a data collection tool. The most meaningful way such data could be collected for a particular speech recognition system would be to categorize the /t/ realizations based on the recognition components of that speech recognition system. However, to make the conclusions obtained from such data of general use, somewhat more general measures should be used. Further, in order to accommodate certain non-categorical hypotheses about phonological variation (i.e. that there is a continuum of possible realizations from canonic-t through d-stop, glottal-stop and flap to deletion) certain continuous measures should be collected in addition to the categorical measurements.

Our philosophy, in relation to the above problems, has been to initially collect data making the categorical decisions by eye. When the point is reached for the initial three speakers that the types of forms realized are known and hypotheses are proposed about their distributions, the categorical decisions made will be semi-automated. (For instance a user may indicate that there is a /t/ in the following phonetic context somewhere in a small region and the machine will search for it and define its (component) durations and take certain continuous measures.

4.4. Results

4.4.1. Distribution of contexts in the spontaneous speech

Up to this point, we have recorded the speakers and collected some data on the observed forms of /t/ in the intervocalic /t/ context (with optional word boundaries). There are several observations to be made regarding the distribution of forms in spontaneous speech (and how that impacts the amount of speech required to be collected for reasonable statistics).

The table below shows the distribution of contexts for each speaker (speakers SH and JK are male, SS is female) based on phonemics generated by the PROSE-2000. Note that below V=Vowel for left contexts. For right contexts V' = stressed vowel, V = unstressed vowel (including reduced vowels unless v is given too), and v = reduced vowels. When not in contrast to V', V refers to all vowels.

DISTRIBUTION OF /T/ ACROSS CONTEXTS (30 minutes of speech)		
Context	Speaker	
	SH	JK	SS
S_TV'	4	11	6
S_TV	5	15	14
ST_V'	3	1	5
ST_V	12	16	17
STV'	0	0	0
STV	10	5	7
STv	12	6	15
V_TV'	32	30	37
V_TV	28	20	23
VT_V'	50	37	40
VT_V	109	58	136
VTV'	20	8	23
VTV	62	99	71
VTv	48	56	65
VT_S	30	24	31
VTS	72	65	97
N_TV'	9	4	11
N_TV	3	7	8
NT_V'	7	4	4
NT_V	18	10	12
NTV'	6	10	8
NTV	11	14	35
NTv	7	6	9
V_TJ	0	0	0
VT_J	26	12	21
VTJ	0	0	0
V_TN	0	0	0
VT_N	6	7	9
VTN	8	3	9
V_TW	0	3	4
VT_W	46	54	39
VTW	1	0	2

47

Ignoring word boundaries the following distributions occur:

DISTRIBUTION OF /T/ ACROSS CONTEXTS			
(Ignoring word boundary)			
(30 minutes of speech)			
Context	Speaker		
	SH	JK	SS
STV		52	59
STS	4	13	7
STW	3	3	5
VTV	258	302	316
VTS	97	86	120
VTW	47	54	45
NTV	58	51	81
NTS	8	13	12
NTW	7	4	3
LTV	10	9	14
LTS	1	1	0
LTW	0	0	0

It is clear from the above data that it will be difficult for realization probabilities to be estimated for many context from spontaneous speech, and a strategy of "going to more general contexts when insufficient data is available for a current context" will have to be adopted, as has been used in the IBM and BBN speech recognition efforts.

4.4.2. Realized /T/ forms for VTV

A small study of intervocalic /T/ for the first 10 minutes of each speakers first session was performed. This study and others like it will be used to guide the construction and the areas of application for the semi-automated data collection tool described above. The data collected (limited to about 10 data points from any given context) was:

REALIZATIONS OF /T/ ACROSS CONTEXTS					
(up to 10 minutes of speech per speaker)					
Context	Delete	Glot	Flap	D-stop+T-rel	Canonic
T_V'					
SH			2	2	
JK	1	2	2		
SS		2	6	1	
T_V					
SH	3		4	2	
JK	3	1	3		
SS			1	5	
_to-word					
SH			1		
JK	3		1		1
SS				1	3
_TV'					
SH					5
JK					10
SS					10
VTV					
SH					
JK	2		4	3	
SS			4	5	2
VT-ate					
SH	3		1		2
JK					3
SS				1	4

This data, which was chosen for preliminary study because it was the easiest to analyze, shows few differences among these three speakers. In general, the "flap" form and the "d-stop+t-release" form are both *heard* as flaps, but speaker SS seems to differentiate the two in final /t/s depending on the stress of the following vowel. SS also seems more likely to asperate the /t/ in the word <to>. This kind of anaysis is continuing.

5. SUMMARY

This paper has covered three activities at SRI, all related to the goal of accommodating within and across speaker variation in speech recognition design. The network and rule manipulation tools are still evolving, and the study of /t/ allophonics across speakers is in progress. Several key issues remain in all of these endeavors, including technical question like how to merge processes and still recover the probabilities of the events, and system issues like how to integrate this knowledge in a lexical access algorithm.

REFERENCES

J.Bernstein, M.Kahn & T.Poza (1985) "Speaker Sampling for Enhanced Diversity", IEEE Proc. ICASSP-85, paper 41.2.

D.Klatt (1976) "Linguistic uses of segmental duration in English", J. Acoust. Soc. Am. 59, pp. 1208-1221.

D.Klatt (1979) "Synthesis by rule of segmental durations in English sentences," a chapter in Frontiers in Speech Communication Research (B.Lindblom & S.Ohman, eds., New York, Academic Press.)

N.Umeda (1976) "Linguistic rules for text-to-speech synthesis," Proceedings of the IEEE 64(4), 443-451.

The Role of Word-Dependent Coarticulatory Effects
in a
Phoneme-Based Speech Recognition System

Yen-Lu Chow, Richard Schwartz, Salim Roucos, Owen Kimball,
Patti Price, Francis Kubala, Mari O. Dunham,
Michael Krasner, John Makhoul

BBN Laboratories
10 Moulton St.
Cambridge, MA 02238

ABSTRACT

This paper describes the results of our work in designing a system for large-vocabulary word recognition of continuous speech. We generalize the use of context-dependent Hidden Markov Models (HMM) of phonemes to take into account word-dependent coarticulatory effects. Robustness is assured by smoothing the detailed word-dependent models with less detailed but more robust models. We describe training and recognition algorithms for HMMs of phonemes-in-context. On a task with a 334-word vocabulary and no grammar (i.e., a branching factor of 334), in speaker-dependent mode, we show an average reduction in word error rate from 24% using context-independent phoneme models, to 10% when using robust context-dependent phoneme models.

1. INTRODUCTION

It is well known that the acoustic realizations of phonemes in continuous speech vary with the phonetic context that the phoneme is in. The resulting coarticulatory effects are most pronounced for immediately adjacent sounds, but have been known to extend to several phonemes away from the observed phoneme. In our continuous speech phonetic recognition work [1, 2], we have modeled these coarticulatory effects by using context-specific phoneme models. In particular, we have defined a unique model for a phoneme in each of its different phonetic environments (as determined by the context of immediately adjacent phonemes). We call this context-specific model a "phoneme-in-context" model. Since this set of models is very large (in principle, the cube of the number of phonemes) we cannot expect to model all of them very well with a limited training set. In general, these models are combined (smoothed) with less detailed (but more robust) models with weights that depend on the amount of training of each model.

In our phonetic recognition experiments we have

observed that the improvement in performance due to using diphone-dependent models of phonemes instead of context-independent models, for example, is smaller when the test vocabulary was different from the training vocabulary – even though the diphones in the test set had occurred frequently in the training set. We hypothesized that contexts beyond the immediate phonetic contexts are important and affect recognition results. This may be a main reason why speech recognition systems that model whole words typically outperform those that use a phoneme model, as long as the amount of training for each word is sufficient and the effects between words are not severe. However, word-based systems cannot easily take into account word boundary effects and are not easily extensible to vocabularies of thousands of words. The problem then is to model phonemes in context to maximize recognition performance on a particular large vocabulary, especially when not all the words in the vocabulary appear often enough in the training set to allow the estimation of robust models.

In this paper, we demonstrate the effects of word-level contexts on recognition performance. We describe a method for incorporating word-dependent coarticulatory effects in a phoneme-based speech recognition system.

The paper is organized as follows. Section 2 gives an overview on the modeling of phonemes in context, accompanied by a description of training and recognition algorithms. Section 3 describes the recognition task domain and the database used for the experiments presented here. Section 4 contains performance figures for using different levels of context, along with a discussion of the results. Finally, Section 5 presents conclusions drawn from this work.

2. FRAMEWORK FOR MODELING COARTICULATION

In a phoneme-based speech recognition system, each

word in the lexicon is decomposed into phoneme subunits, each of which can be modeled by a single HMM. We have previously argued, however, that performance can be improved by taking into account immediate phonetic context. In this paper we have extended this concept to incorporate the greater detail of word-dependent contextual models. Each phoneme within a word is, in principle, modeled as depending on the word in which it occurs. However, if the word has not been observed a sufficient number of times, information about the acoustic realization of the same phoneme in similar phonetic environments can be generalized from other words in the training set. In fact, we combine these several hierarchical models of a phoneme (word-dependent, triphone context, left and right context, no context) with continuous linear weights that depend on the number of occurrences of each type of unit, the location within the phoneme, and the relative importance of each unit on the acoustic realization of the phoneme.

Training and Recognition

The different context-dependent models together form a unique expanded HMM network for each phoneme of each word in the lexicon, where the less detailed models are shared across different words. These models are trained jointly using the Forward-Backward algorithm to obtain the maximum likelihood estimate of the HMM parameters for all the different context models given the training data.

Once the training is completed, we precompute a single model for each word in the lexicon from a complete set of these phoneme-in-context models acquired during training, given the pronunciations for the word. We combine all relevant context models that were observed with appropriate weights to obtain a robust model for each phoneme in the context of the particular word.

The combined model should achieve the high performance of word-based recognition systems for words that have been observed sufficiently, while allowing reasonable performance for less frequent words or words that have never been observed.

3. EXPERIMENTAL CONDITIONS

In this section, we describe the database and the task domain for the recognition experiments described below

Database

The vocabulary used in this study was from a 334-word electronic mail task. A total of 400 different sentences were generated covering 250 words of the vocabulary. (200 words of the vocabulary and 100 of the sentences were supplied by CMU.) The sentences were each recorded by three male speakers in sessions of 100 sentences, separated by a few days. The first three sessions were designated as training data, and the last as test material. The total duration of the training material was thus about 15 minutes for each speaker. The test material used in the experiments below included 30 of the test sentences, with a total of 187 word tokens covering 80 different words. Each test word occurred at least once in the training set.

A dictionary of phonetic pronunciations was constructed for this 334-word vocabulary without listening to either the training or test material, but by trying to account for the most frequent phonological variations for each word. The average number of different pronunciations per word was 2. Word boundary phonological variations were *not* included.

Analysis

The sentences were read directly into a close talking microphone in a natural but careful style in a normal office environment. The input speech was lowpass filtered at 10 kHz and sampled at 20 kHz. Fourteen Mel-frequency cepstral coefficients (MFCC) were computed every 10 ms on a 20 ms analysis window. Some of the training data was used with a k-means clustering algorithm to produce a representative set of MFCC vectors. The k-means clustering was found to result in slightly better performance than a nonuniform binary clustering procedure. These experiments were performed using a codebook size of 256 MFCC templates. Each MFCC vector in the training and test sets was then classified using vector quantization (VQ) [3], as one of the 256 template vectors. To save computation, strings of up to 3 identical vector codes were compressed to 1 observation. (This simple variable frame rate scheme was found not to affect performance.)

Training

To obtain the necessary initial estimate for the probability density functions (pdfs) for each state of the phonetic HMM we use a bootstrapping technique A separate passage (5 minutes of speech of a different vocabulary) spoken by one of the talkers was carefully labeled, indicating the beginning frame of each phoneme The hand-labeled speech was then quantized using the VQ codebook for each particular talker in the experiment. Normalized histograms of the observed

vector-quantized spectra for each phoneme were computed from the labeled data to form an initial estimate of the pdf for that phoneme for that talker. This bootstrapping technique of using a single talker's speech as an initial estimate for all talkers seems to work quite well. All the pdf's for the different states in the HMM for a phoneme are set to this initial estimate. Finally, all the pdfs for the context-dependent models of a phoneme are set equal to the single, context-independent model of that phoneme.

The 15 minutes of training data per talker is transcribed with the sequence of words spoken (no time labels and no phonetic labels). The training data is then processed with five passes of the Forward-Backward algorithm. In the cases where context-dependent models of the phonemes are used, the training algorithm maintains separate models for each observed phonetic context.

Prior to recognition, word models are precomputed for each word in the vocabulary from the appropriate phoneme-in-context models with weights depending on the number of occurrences of each model and the position within the phoneme (as used in training).

Recognition

The recognition algorithm used is a time-synchronous procedure [2], which attempts to find the sequence of words that are most likely given the observed sequence of vector quantized spectra in a test utterance. At present, no grammar is used, tnus making the effective branching factor or perplexity equal to the vocabulary size (334).

The recognized sequence of words is then compared automatically to the correct answer to determine the percentage of correct, deleted and inserted words. Word substitutions and deletions are tabulated as errors, while insertions are counted separately.

4. EXPERIMENTAL RESULTS

In this section we present results on several word recognition experiments on continuous speech. As described in the previous section, the results were produced for the following set of conditions 3 speakers, speaker-dependent, 334-word lexicon, electronic mail task, no grammar, 15 minutes of training, and 30 test utterances totaling 187 words.

Table 1 gives a detailed description of the various system configurations for the different experiments.

system name	word models are constructed using
PH	context-independent phoneme models
W	only word-dependent phoneme models, regardless of whether training is sufficient for the word
PH+W	linear interpolation of context-independent and word-dependent phoneme models
PH+L+R	linear interpolation of context-independent, left-context-dependent and right-context-dependent phoneme models.
PH+L+R+W	linear interpolation of context-independent, left-context-dependent, right-context-dependent, and word-dependent phoneme models.

Table 1: Different System Configurations for Word Recognition.

Table 2 shows word recognition error rates obtained for many different configurations of the system. A complete set of results was obtained for three speakers, RS, FK, AW. Also shown is the average error rate for the three speakers, and the difference in error rates from the best to the worst speaker for each system. The word insertion rates are not given for each speaker and system below. However, for systems PH and W, the insertion rate ranges from 5-10%, while for the other systems, which use combinations of models, the range is from 2-4% in all cases.

	RS	FK	AW	AVG	Best minus Worst Error Rate
PH	15	25	32	24	17
W	11	17	14	14	6
PH+W	8	11	12	10	4
PH+L+R	10	12	14	12	4

Table 2. Experimental Results

The first experiment (PH), with word models derived from context-independent phoneme models, constitutes

our baseline result. The word recognition error rate is 24% averaged across the three talkers. The second experiment (W), using word-dependent phoneme models only, resulted in an average error rates of 14%. In the remaining experiments, the model of each word was constructed from combinations of several models with appropriate weights, in an attempt to improve performance over using the word-dependent model by itself. The error rates are as follows. PH+W - 10%, PH+L+R - 12%

From the results given above, we make the following observations. First, the systems that model coarticulatory effects clearly result in better recognition performance. For example, system W achieves significantly better performance than system PH. This result is due to the facts that each word in the test set has been observed at least once in the training, and that word-dependent coarticulatory effects are important. Although some words are poorly trained, the overall performance is improved. Note that for larger vocabularies, many words would not occur in training, making this system (W) inappropriate. A system that uses a subword context-dependent model will be necessary. Second, the systems that use less detailed models to smooth the highly context-dependent models result in better performance than those that attempt to use the context-dependent model by itself. For example system PH+W outperforms system W. Third, the range in performance across the three speakers (17%) is large for the context-independent (PH) system. We conjecture that this is due to a difference in the degree of coarticulation present. Speaker RS has been subjectively judged to speak more carefully than the other two speakers. However, the range in performance for the context-dependent systems (4-6%) is greatly reduced - a desirable attribute. We believe this behavior is due to the fact that these systems are better able to model coarticulation.

As as side note, we tried combining all four models (PH+L+R+W) in a single experiment, but found that performance did not improve over the PH+W system. We presume that this is due to the fact that most words in the test set were well trained.

5. CONCLUSION

In conclusion, we have made two major extensions to concepts that were introduced in our previous work. First, we have extended the use of context-dependent phoneme models to the case of continuous speech word recognition. Second, we have extended the phoneme-in-context models to account for word-dependent coarticulatory effects. Our method, based on a context-dependent phonetic hidden Markov model, automatically uses information about adjacent phonetic context only to the extent that it has seen examples of that context in training, and combines this information with less context-specific models for the phoneme. Systems that make use of robustly combined (smoothed) word-dependent models of the phoneme are demonstrated to have the best performance.

References

1   R.M Schwartz, Y Chow, S. Roucos, M. Krasner, and J. Makhoul, "Improved Hidden Markov Modeling of Phonemes for Continuous Speech Recognition", *IEEE Int Conf Acoust., Speech, Signal Processing*, San Diego, CA. March 1984, pp. 35.6.1-35.8.4.

2   R.M Schwartz, Y L Chow, O.A. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, March 1985, pp 1205-1208, Paper No. 31.3.

3   J Makhoul, S Roucos, and H Gish, "Vector Quantization in Speech Coding", *Proc. IEEE*, Vol 73, No. 11, November 1985, pp. 1551-1588. Special Issue on Man-Machine Speech Communication

RECOGNITION PERFORMANCE
AND
GRAMMATICAL CONSTRAINTS

O. Kimball, P. Price, S. Roucos, R. Schwartz, F. Kubala,
Y.-L. Chow, A. Haas, M. Krasner, J. Makhoul

BBN Laboratories. Inc.
10 Moulton Street
Cambridge, MA 02238

ABSTRACT

We describe the integration of grammatical with acoustic knowledge sources in the BBN continuous word recognition system, and the resulting effects on performance. This combination decreases the total number of insertions, deletions and substitutions by a factor of more than 6 compared to the system with no grammatical constraints, and yields a word accuracy of better than 98%. We show that constraining the set of possible word sequences can improve performance, even when the amount of training per lexical item remains fixed. In addition, we address the issues of estimating from limited data the degree of constraint imposed by a grammar and the importance of incorporating acoustic similarity in such measures.[1]

grammatical constraint are perplexity and branching factor; decreasing these characteristics of a grammar should lead to improved performance. We shall discuss how these measures can be estimated when only a small set of representative sentences are available.

In the following section we describe our recognition system. In section 3, we describe a set of experiments designed to demonstrate the relationship of performance to branching factor when the amount of training per item remains constant. We then address the issue of estimating degree of grammatical constraint from limited data (section 4). In section 5, we describe the incorporation of various grammars in our recognition system and the resulting effects on performance.

1 INTRODUCTION

In this report we describe the development and use of various finite state grammars in the BBN continuous speech recognition system. In particular, we investigate the relationship between recognition performance and the degree of constraint imposed by a grammar. We feel that understanding such relationships is crucial to evaluating how well specific techniques of linguistic modeling can be generalized to larger and more complex tasks.

It is well known that recognition performance improves as vocabulary size decreases. Similarly, when syntactic and semantic information are used to reduce the number of words that can legally follow a given sequence of words, a recognizer is expected to make fewer errors. Two related measures of this type of

2 THE SPEECH RECOGNITION SYSTEM

The speech recognition system consists of a feature extraction stage, an acoustic scoring and a linguistic scoring. The feature extraction stage computes the short-time spectral envelope every centisecond and represents it by 14 Mel-warped cepstral coefficients. A vector quantizer discretizes the spectral envelope to one of 256 spectral templates using Euclidean distance. The sequence of discrete spectra is used to compute the likelihoods of all possible hypotheses in the acoustic and linguistic scoring modules. Recognizing an input utterance involves finding the sequence of words $w_1 \ldots w_m$ that maximizes

$$P(x_1 x_2 \ldots x_n \mid w_1 \ldots w_m) \, P(w_1 \ldots w_m)$$

where $x_1 \ldots x_n$ is the sequence of quantized spectra and $w_1 \ldots w_m$ is a sequence of words. The first term, the acoustic score, is derived from a hidden-Markov model (HMM) for each word. The second term, the

linguistic score, is, in principle, a model of the expected syntax and semantics. This term includes a model of duration (longer sequences are less likely), and a grammatical score. At present, due to limited data, the grammatical score is simply set to 1 for sentences allowed by the grammar and to 0 otherwise.

The dictionary used was developed and made available to us by the speech group at Carnegie-Mellon University. We expanded it (from about 200 words) to 334 words in order to fill out categories that were represented in the original version. In particular, our version includes all months, all days of the week, possessives for all proper nouns and plurals for all other nouns, and cardinals and ordinals to cover numbers up to 999.

The training for our system was on 300 sentences (about 15 minutes) for each talker. These sentences were syntactically and lexically based on 100 example sentences also provided by CMU. We reserved the set of 100 sentences for testing. The sentences were designed to be representative of human-machine interaction in an electronic mail task, referred to as the Email task.

Our word models are phonetically based and capture the acoustic coarticulatory effects within a word to the extent that they can be estimated reliably from available training data. In short, to obtain robust estimates of the transition and output distributions of the HMM for a phoneme-in-context we use a weighted average of the parameters of models with varying amount of context. The details of these word models are discussed in [2].

The linguistic model, which computes the *a priori* probability of a word sequence, uses one of two types of models for the language. The first model has no grammar and allows any word sequence. In this case, the probability of a word sequence is determined by its length.

$$P[w_1 \cdots w_k] = c\, a^{-k}$$

where a is just an insertion penalty that is chosen empirically to control the insertion rate of the recognizer output and c is a normalizing constant. The second language model is a finite state automaton. We describe in a later section how we generated the finite state grammars from a small corpus of sentences. At present, sentences are either accepted or rejected as grammatical depending on whether the automaton parses them or not. Given sufficient data to determine the likelihood of different word sequences, the paths of the automaton could be modified to impose probabilities on sentences of the grammar.

3 RECOGNITION ACCURACY AND BRANCHING FACTOR

It is well known that recognition performance improves with smaller vocabulary size, with or without grammatical constraints. The improved performance may stem from two factors. (1) the smaller set of elements that need to be distinguished, and (2) the greater amount of training that can be devoted to each of the items. As vocabulary size increases, comparable training becomes more difficult. Since our goals involve increasing vocabulary size, we felt it was important to establish that the first of the above factors alone, i.e., smaller vocabulary size (which can be simulated by using a grammar), is sufficient to improve performance without increasing the amount of training per lexical item. Further, we would like to investigate the relationship between performance and constraints such as vocabulary size or grammaticality. A set of experiments was designed to simulate the effect of grammatical constraints over a range of branching factors. This was done by restricting the set of lexical items to the words appearing in a given test sentence plus additional words selected randomly from the dictionary until the total number of words is equal to the desired branching factor.

3.1 Methodology

We investigated branching factors of 10, 20, 50, 100, 200, and 334. The last figure includes the entire dictionary. Performance was assessed for the task of recognizing 30 of the 100 test sentences, described earlier, as produced by three male talkers. Since we had previously made changes in our system based on recognition of these 30 sentences, we repeated the experiment for the smallest and largest branching factors on the 70 previously unused sentences. Since performance at these points for the new sentences did not differ greatly from the results based on the 30 sentences (performance was actually about 1% better on the new sentences), we present the results based on the 30 sentences.

In order to achieve comparable statistical significance across the tests at various branching factors (BF), that is, to adequately sample the dictionary for each, we increased the number of repetitions for experiments at lower BF. BF of 10 was repeated at least 10 times per talker per sentence, BF 20 (11 times), BF 50 (6 times), BF 100 (3 times), BF 200 (twice) and BF 334 (once).
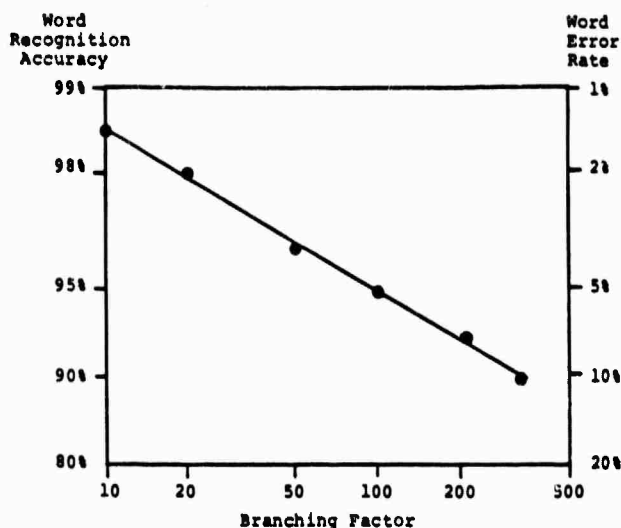
Figure 1: Performance and Branching Factor. Plotted is word accuracy, (substitutions + deletions) divided by the total number of words in the test sentences, averaged across 3 male speakers, as a function of branching factor.

3.2 Results and Discussion

Figure 1 shows the error rate averaged across the 3 talkers' productions of the 30 test sentences. Performance is plotted as a function of branching factor on a log-log scale. It is seen that performance increases (linearly on this scale) with smaller branching factors. word accuracy improves from about 90% for the full dictionary to about 98.5% for the branching factor of 10. As mentioned earlier, performance on the remaining 70 test sentences was about 1% better for branching factor of 10. The repetitions of the experiment allow us to sample the effects of various choices of vocabulary items, but not the effects of variability in articulation. In fact, our entire set of errors for the branching factor of 10 correspond to one or two words produced by each talker. Given this distribution of errors and the difference between the percentage of errors on the two sets of sentences, we conclude that 30 test sentences (187 words per talker) are not sufficient to reliably estimate performance in this case. The experiment has, however, confirmed our hypothesis that reduction of the number of allowable words is sufficient to improve performance without

increasing training, and we feel that the methodology may prove useful for estimating the performance of a recognition algorithm on tasks differing in the complexity required of the grammar. In order to quantify this complexity, we present several methods for estimating the amount of constraint imposed by a grammar.

4 ESTIMATING GRAMMATICAL CONSTRAINT

When recognition is performed without a grammar, the set of possible outcomes is the set of all possible combinations of the lexical items. The role of a grammar is to disallow some of those combinations. This means that at any point the grammar has to choose not from the entire set of lexical items, but from a smaller set. By reducing the legal possibilities the grammar imposes a constraint which makes the recognizer's task easier. How does one measure the constraint imposed by the grammar? One would like to average the number of choices at various points and weight them according to how likely they are to occur. Such a measure, based on the information theoretic concept of entropy, exists and is called "perplexity" [1]. For a deterministic finite state automaton we define its entropy, H, by

$$H = \sum p(i) \, h(i)$$

where $p(i)$ is the probability of node i, and $h(i)$ is the entropy of the set of choices emanating from that node. The perplexity, Q is

$$Q = 2^H$$

The perplexity of a grammar is determined by the network connectivity and the probability assignment of the different transitions. In our case, the network connectivity is determined by the types of linguistic phenomena captured in a particular grammar. The probability assignment of the transitions is, however, more difficult. The basis for our grammar was a set of 100 sentences intended to represent rather than to define the language. In fact, many different grammars can be built to cover all or most of these sentences while differing greatly in the number and type of additional sentences covered, and, more importantly, differing in their perplexity. The problem now becomes the estimation of perplexity given a set of "representative" sentences. We propose three methods.

55

The first is the maximum perplexity of a finite language [6] which is obtained by solving for the positive root x_0 of

$$\sum_{k=1}^{l_{max}} N_k \, x^{-k} = 1$$

where N_k is the number of sentences of length k in the language, l_{max} is the length of the longest sentence in the language, and x_0 is the desired maximum perplexity.

A second measure, which we will call the uniform branching estimate of perplexity, is obtained by assuming all transitions from a node in the grammar to be equally likely.

The third measure, called test set branching factor, uses the set of test sentences to estimate the average branching factor encountered by traversing the FS network along the paths corresponding to each sentence. We use the geometric mean of the number of branches at each node over all the test sentences as an estimate of task perplexity.

All the above measures ignore the acoustic similarity of the words, an important factor. Measures including this factor have been proposed, see, for example, [3].

5 RECOGNITION ACCURACY AND GRAMMATICAL CONSTRAINTS

In this section, we compare recognition performance using grammars differing in the degree to which they constrain the set of allowable word sequences. We began with a grammar designed to cover a structural subset of the Email sentences, the commands. A goal of this grammar was to maximize coverage of these sentences plus logical extensions suited to the Email task environment. Equally important in the design of this grammar was the minimization of "over-generation", i.e., the generation or acceptance of many ungrammatical sentences.

Our interest in grammars is broader than simply improving performance on a given task. In addition, we would like to investigate the trade-off in performance versus over-generation, and to estimate performance on more difficult tasks, i.e., tasks requiring a larger number of choices at various points in the grammar. We therefore designed a second grammar for the commands, a grammar with greater perplexity. Similarly, we designed two grammars differing in perplexity for the entire set of sentences (commands as well as questions).

5.1 Integration of Grammatical Constraints in the Recognition System

We approached the implementation of a grammar in our recognition system in two steps. First we created a description of the Email task language in a modified context-free notation. This description was based on the 100 sentences mentioned earlier, and was designed to capture generalizations of the linguistic phenomena found in them. Second, we created tools that transformed this description into structures in our recognizer that provide the corresponding grammatical constraint. These tools provide us with a general facility for capturing in our recognition system an approximation of any language expressible in context-free rules. We chose to implement the constraints in the recognition system in the form of a finite automaton (FA) similar to those described in [4] and [1].

At the first stage in generating a grammar, we use a context-free notation augmented with variables in order to simplify the process of describing a language. For example, this notation would allow a rule that says a noun phrase of any number can be replaced by an article and a noun of the same number, whereas ordinary context-free notation would require two rules that are identical except that one would be for singular number and the other for plural.

Our system first translates the augmented notation into ordinary context-free rules and then constructs a FA based on these rules. While it is true that context-free grammars can accept recursive languages which finite automata cannot, finite automata can approximate recursion by setting upper limits on the number of levels of recursion allowed. Such an approximation is reasonable for most task languages, since spoken sentences do not ordinarily use more than a few levels of recursion.

In our recognition system, the automaton is used as follows. Associated with each transition in the FA is a hidden-Markov word model that is used to compute the probability of a spectral sequence given the occurrence of the word at that place in the grammar. The recognition algorithm with this grammar is only slightly different from the version of the algorithm that allows any sequence of words [2]. For each 10 ms frame of the input speech, the scores for all the word models in the FA network are updated according to a modified Baum-Welch algorithm. The score for the start state of the FA is unity and the score for every other FA state is simply the maximum of all the word model scores that enter the node along FA transitions. This state score, in turn, is propagated to the beginning of all the word

models on transitions leaving the state, to be used as the new initial score for those models. In this way the recognizer only considers grammatical sequences of words. Maintained throughout this scoring process are traceback pointers that indicate for each state and each time the word model that produced the best score to enter the state. Once an utterance is thus processed, it is a simple matter to follow these pointers back through the network to find the highest scoring sequence of words.

One potential difficulty with a FA grammar for recognition stems from the fact that, ordinarily, computation is proportional to the number of transitions in the FA. This number can become quite large for complex languages. However, in our experience with grammars for the Email task, a simple time-synchronous search with pruning [5] effectively reduces the computation to less than that for the algorithm that does not use a grammar, without affecting performance.

5.2 Description of the Grammars and Methodology

We compare here the effects on performance of grammars differing in which set of sentences they are intended to cover (the full set of test sentences or the commands only) and along a dimension we call tight-loose, which refers to an estimate of how much over-generation is produced by the grammar. "Tight" grammars have very little over-generation (generation of sentences that are considered ungrammatical) and, because of these tighter constraints, tend to have fewer choices at various points in the grammar, i.e., smaller perplexity. "Loose" grammars, on the other hand, have a great deal of over-generation and greater perplexity (larger sets of choices at various states). The loose grammars developed here are loose in that, for example, no number, tense, case or semantic agreement is required.

The grammars we have investigated so far include a tight and a loose grammar for commands (COM-T and COM-L, respectively) and a loose grammar that covers both commands and questions (SENT-L). In addition, we have used another grammar that is tighter than SENT-L (and hence is called SENT-T), but only in aspects that would otherwise put into similar grammatical distribution large sets of minimal pairs. For example, singular versus plural nouns, the cardinals versus ordinals, or verb tenses all involve large sets of acoustically similar items. This fact can pose a problem for recognition if the grammar allows many sequences in which one member of the pair can be substituted for the other.

On the other hand, distinguishing verbs on the basis of which objects they take reduces perplexity without necessarily reducing the number of acoustically similar competing words.

Table 1 shows the relevant attributes of the grammars investigated. For comparison, the results for no grammar (the trivial grammar that allows any lexical item to occur anywhere) are also included. The table includes: the number of arcs (a rough measure of size, and is related to computation time), the three estimates of perplexity (Maximum Perplexity, Test Set Branching Factor, and Uniform Branching). This table also shows the number of words and number of sentences on which each grammar was tested, and the performance for each. Word accuracy here is computed as the sum of all errors (insertions + deletions + substitutions) divided by the sum (total words + insertions). Sentence accuracy is also included in order to show that a few percentage points difference in word accuracy can result in much larger differences in the number of correctly recognized sentences a number that is no doubt very important to potential users.

Since we had used 30 of the 100 test sentences in previous experiments and modified our system as a function of those results, we used only the subset of 70 remaining sentences for the performance figures reported here. In order to compare the tight and loose versions of the grammars, performance was assessed using the intersection of the sentences parsed by each grammar. Results are based on using the phone-left-and-right word-model discussed in [2].

5.3 Results and Discussion

Figures 2a (commands only) and 2b (commands and questions) show graphically the word accuracy figures of Table 1 associated with each grammar. Performance is plotted as a function of the perplexity estimates used. As can be seen these grammars differ in their effects on performance. Further, when two grammars that cover the same set of sentences are compared (COM-T versus COM-L or SENT-T versus SENT-L), the more constrained grammar has significantly better word accuracy than the less constrained one. tightening of the command grammar improved performance from 95.5% to 98.4%, tightening of the sentence grammar improved performance from 96.2% to 98.2%. Word accuracy, again, includes as errors all insertions, deletions and substitutions. Further, it appears that grammatical constraints that take into account acoustic similarity

TABLE I

Properties of the Grammars

GRAMMAR	COM-L	COM-T	SENT-L	SENT-T	NONE
Number of arcs	838	7167	2547	3771	
Maximum Perplexity	58	19	75	60	334
Test Set Branching	40	18	47	31	334
Uniform Branching	19	9	22	19	334
Words in test set	183	183	438	438	492
Sentences in test set	27	27	63	63	70
Sentence accuracy	72.9%	90.1%	80.5%	90.25	36.7%
Word Accuracy	95.5%	98.4%	96.2%	98.2%	88.6%

Comparison of the various grammars used for the commands (tight coverage, COM-T; loose coverage, COM-L) and the commands plus questions (tight coverage, SENT-T; loose coverage, SENT-L). Word accuracy here is computed as (insertions + deletions + substitutions) divided by (total words + insertions).
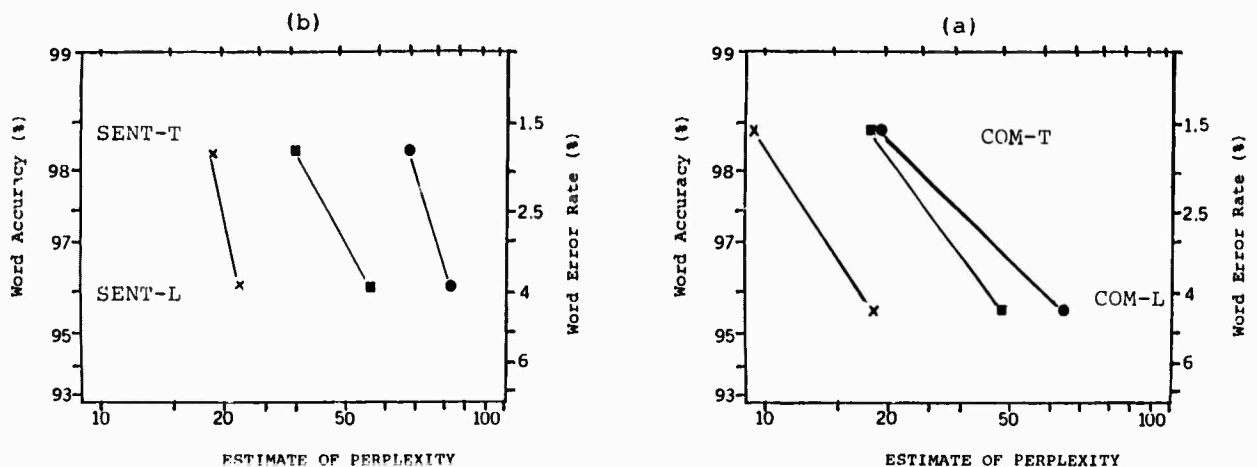
Figure 2: Performance with Grammars. Plotted is performance, (insertions + deletions + substitutions) divided by (number of words + insertions), as a function of perplexity as estimated by the uniform branching assumption (X), the test set branching factor (squares), and the maximum perplexity (circles). (a) The tightly constrained command grammar (COM-T) and its loose counterpart (COM-L). (b) The tightly constrained sentence grammar (SENT-T) and its loose counterpart (SENT-L), which considers acoustic similarity.

58

improve performance more than those that do not for comparable estimated perplexity. the SENT-L grammar improves performance more than its estimated perplexity would predict if acoustic similarity had not been an important factor.

An analysis of the recognition errors using these various grammars reveals that, in general, acoustically similar items are confused. it does not appear that function words are more often involved in the errors than content words. A large percentage of our errors (32% for SENT-T) involve "the" and "a", which happen to be function words. However, no other function words show this pattern. We believe that "the" and "a" show up more often in the errors NOT because they are function words, but because they are (1) acoustically similar, (2) have similar grammatical distributions, and (3) are very frequent words in these sentences. Assuming that we cannot change their acoustic similarity or their lexical frequency, improving performance on these words requires a more constrained specification of their distribution in the linguistic model. It is possible that semantic, pragmatic or discourse models could separate the two distributions, given a well-defined task environment.

6 CONCLUSIONS AND FUTURE RESEARCH

We have implemented and tested methods of combining grammatical and acoustic knowledge sources in our recognition algorithm. We find that the use of grammatical constraints can decrease the error rate by a factor of more than six. This result corresponds to a word accuracy (counting all insertions, substitutions and deletions as errors) of more than 98% for the Email task. Reducing the number of words considered by the recognizer boosts performance, even when the amount of training per word is fixed. We have presented various estimates of grammatical perplexity and shown that performance improves as estimated perplexity decreases for a given task. Our experience with a grammar that focuses only on syntactic constraints in acoustically confusable portions of the grammar demonstrates the importance of acoustic similarity in predicting performance accurately and in improving recognition performance.

References

1. Y.L. Chow, R.M. Schwartz, S. Roucos, O.A. Kimball, P.J. Price, G.F. Kubala, M.O. Dunham, M.A. Krasner, and J. Makhoul, "The Role of Word-Dependent Coarticulatory Effects in a Phoneme-Based Speech Recognition System", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tokyo, Japan, April 1986.

2. L.R. Bahl, F. Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 2, March 1983, pp. 179-190.

3. M.M. Sonhdi and S.E. Levinson, "Computing Relative Redundancy to Measure Grammatical Constraint in Speech Recognition Tasks", *IEEE Int. Conf Acoust., Speech, Signal Processing*, Tulsa, OK, April 1978, pp. 409-412

4. R.G. Goodman, *Analysis of Languages for Man-Machine Communication*, PhD dissertation, Carnegie-Mellon Univeristy, May 1976.

5. S.E. Levinson, "Structural Methods in Automatic Speech Recognition", *Proc. IEEE*, Vol. 73, No. 11, November 1985, pp. 1625-1650. Special Issue on Man-Machine Communications

6. B.T. Lowerre, *The Harpy Speech Recognition System*, PhD dissertation, Carnegie-Mellon University, 1976.

IMPLEMENTATION OF CONTINUOUS SPEECH RECOGNITION ON A BUTTERFLY™ PARALLEL PROCESSOR

Lynn Cosell, Owen Kimball,
Richard Schwartz, Michael Krasner

BBN Laboratories
10 Moulton St.
Cambridge, MA 02238

ABSTRACT

This paper describes the implementation of a continuous speech recognition algorithm on the BBN Butterfly™ Parallel Processor. The implementation exploited the parallelism inherent in the recognition algorithm to achieve good performance, as indicated by execution time and processor utilization. The implementation process was simplified by a programming methodology that complements the Butterfly architecture. The paper describes the architecture and methodology used and explains the speech recognition algorithm, detailing the computationally demanding area critical to an efficient parallel realization. The steps taken to first develop and then refine the parallel implementation are discussed, and the appropriateness of the architecture and programming methodology for such speech recognition applications is evaluated.[1]

INTRODUCTION

This paper describes research to investigate the uses of parallel computation for continuous speech word recognition. Our goal in this work is to determine the extent to which continuous speech recognition algorithms can make use of parallel processing to achieve real time speeds. Our approach has been to develop parallel versions of an existing recognition algorithm on BBN's Butterfly Parallel Processor. Additionally, this work provided a chance for us to learn about the Butterfly Parallel Processor and parallel algorithms in general, we approached the project with no prior experience on this or any other parallel machine. The outline of this paper is as follows. section 2 describes the Butterfly, section 3 explains the BBN word recognition algorithm, section 4 describes the single processor version of the program, section 5 presents the parallel programming methodology used, section 6 explains the parallel versions and the final section contains results and future work.

BUTTERFLY

The Butterfly Parallel Processor [1] is composed of multiple (up to 256) identical nodes interconnected by a high-performance switch. Each node contains a processor and memory. The switch allows each processor to access the memory on all other nodes. Collectively, these memories form the shared memory of the machine, a single address space accessible to every processor. All interprocessor communication is performed using shared memory. From the point of view of a program, the only difference between references to memory on its local node and memory on other nodes is that remote references take a little longer to complete. Typical memory referencing instructions accessing local memory take about 2 microseconds to complete, whereas those accessing remote memory take about 5 or 6 microseconds. The speeds of the processors, memories and switch are balanced to permit the system to work efficiently in a wide range of configurations.

The Butterfly has a number of interesting architectural characteristics. It is a multiple instruction multiple data stream (MIMD) machine where each processor node executes its own sequence of instructions, referencing data as specified by the instructions. The MIMD architecture permits a variety of approaches to programming the machine, making it suitable for many applications. Processor Nodes are

tightly coupled by the Butterfly switch. Tight coupling permits efficient interprocessor communication and allows each processor to access all system memory efficiently. The Butterfly Parallel Processor is expandable to 256 Processor Nodes and each processor node added to a configuration contributes processing power, memory, switch bandwidth, and I/O capacity. As a result, processing, memory, communication and I/O capacity all grow with the size of the configuration.

Processor Nodes are all basically identical in this architecture. As a result, every processor is capable of performing any application task. This uniformity simplifies programming since programmers need not concern themselves with allocating certain tasks to specific processors. Each Processor Node contains a Motorola MC68000 microprocessor, from 1 to 4 MBytes of main memory, a co-processor called the Processor Node Controller, memory management hardware, an I-O bus, and an interface to the Butterfly switch. Butterfly Processor Nodes are currently being manufactured with 68020 microprocessors and a floating point co-processor to replace the single 66000 processor.

The Butterfly switch uses packet switching techniques to implement high performance, reliable, and economical interprocessor communication. The switch is a collection of switching nodes configured as a "serial decision" network. There is a path through the switch network from each processor node to every other Processor Node. The name derives from the connection of switch nodes, which resembles an FFT "butterfly" flow graph.

The particular machine that was used for development in this project was a 16 processor machine with a 68000 microprocessor and 1 Mbyte of memory on each Processor Node. As such, it did not have hardware support for floating point arithmetic.

WORD RECOGNITION ALGORITHM

The problem of speech recognition requires that we map an analog signal onto a sequence of words that comprise sentences. However, the identification of particular speech sounds (phonemes) from this signal is made difficult due to the variability that occurs in speech production. This variability is due to the effect of neighboring speech sounds (coarticulation), and a variety of other effects that all combine to make the same speech unit appear differently each time it occurs.

Our recognition algorithm is based on the explicit modeling of variability in speech through the use of probabilistic Hidden-Markov Models (HMMs) of phonemes in various phonetic contexts [2]. Figure 1 illustrates the hidden-Markov model of a phoneme used in our system. The three large open circles are states associated with acoustic events corresponding roughly to the beginning, middle and end of a phoneme. The small filled circles are the "initial" and "final" states and do not produce output symbols. There is associated with each pair of states a transition probability $a(j|i)$ which is the probability of going to state j given that the process is in state i. The arrows between states indicate the allowed (probability nonzero) transitions. Unlike a Markov chain, in which each state has associated with it a single output, each HMM state has an output probability density function (pdf) $P(x|i)$ that gives the probability of each possible output symbol x, given that the process is in state i.

Rather than use actual segments of the speech signal as output symbols, we can represent the speech signal as a sequence of spectra that occur at discrete time intervals. Furthermore, each of these spectra can be approximately characterized as one of a small number of spectral types (256 in our system), which are determined using a clustering procedure. The spectral characterizations, each a single number, then become the possible output symbols. Given this phoneme model, words can be modeled in turn as concatenations of phoneme HMMs that have been modified to take into account the contextual effects of the word.

One approach to understanding HMMs is to imagine them in a synthesis role, where they are used to produce spectral sequences. Starting from the initial state of the model, we randomly choose the next state according to the transition probabilities on the arcs leaving the initial state. Whenever the process is in an acoustic state (the open circles in the diagram), we randomly pick an output symbol according to the state's output pdf. As the process moves from state to state in this manner, it produces a sequence of symbols (speech spectra) as output.

The recognition problem can be viewed as the inverse of the synthesis problem: given a sequence of input spectra and a set of models of all possible words, we wish to find the sequence of models that is most likely to have produced the spectral sequence. An adaptation of the Viterbi algorithm [3] solves just this problem, and is the basis for our recognizer.

The basic recognition algorithm finds the path through the states that is most likely to have produced the spectral sequence to be recognized. The algorithm does this by finding the best path to every state at every time given that the path to be previous state was also "best". Each step along these paths has associated with it a "score", which reflects the probability of the step given the spectral sequence and the transition probabilities of the model. The scores are accumulated along the paths, so that, at every time, the best path to any state has a single score. The best path to a particular state, S, at time t, is determined by considering all possible predecessor states (states which have transitions to S) at time $t-1$ and the best path to each of these. The score for a path to S is then the combination of the path score to the predecessor state and the score for the step from the predecessor state to S. The best of these scores indicates the best path to S.

The central computation in the algorithm is, for each time interval, update the scores for all states. Figure 2 schematically illustrates the scoring procedure for a single state in a word. In this figure, the score for state n at time t (the $\alpha_n(t)$ in the lower right corner) is being computed based on the scores for three states computed at time $t-1$ (the three circles on the left of the figure) each multiplied by the corresponding transition probability of going to state n. The new score for state n is just the maximum of the entering scores multiplied by the probability $p(x|n)$ of the spectrum x_t at time t, at state n.

This scoring procedure is applied to all states in each word for all time frames in the utterance. In the general Viterbi decoding algorithm, the best path to each state would also be maintained. However, in the speech recognition system, this detailed a record is not required. It is sufficient to know the word and the time frame in which the word began. The scores computed for terminal states (ends of words) are special. They

are compared and only the maximum terminal score over all words is saved, along with the word that produced it and the start time for the word. The largest terminal score for a time frame is used as the score for all initial states in the next time frame. In addition, the best score at any state at a particular time is found and used to normalize all scores in the next time frame. This normalization factor (NF in Figure 2) prevents arithmetic overflow. The determination of the normalization factor is indicated at the right hand side of Figure 2. The current state score, $\alpha_n(t)$, is compared against the largest state score, $\alpha_{Best}(t)$, encountered so far for the current time frame, and replaces it, if appropriate.

When all the time frames in the utterance have been processed in this way, the best sequence of words, called a "theory", for the content of the utterance can be determined. The theory for the utterance is determined by backtracking. The maximum terminal score at the end of the utterance specifies the last word of the utterance. The start time of this last word is the end time for the previous word, so the maximum terminal score at this time indicates which word should be selected as the second-last word in the theory, and so on, back to the beginning of the utterance.

SINGLE PROCESSOR IMPLEMENTATION

The first step toward a parallel implementation was to bring up the speech recognition program on a single processor of the Butterfly Parallel Processor. The existing VAX implementation, written in C, depended on the file system to store the large amounts of data which included the transition probabilities and the pdfs for the word models. This data totaled 1.5 Mbytes. Storing this amount of data required using some parallel memory management techniques to allocate shared memory on multiple nodes.

The VAX (and the first Butterfly implementation) used floating-point arithmetic. Because floating-point arithmetic is performed in software in our Butterfly Parallel Processor it seemed likely that the floating-point arithmetic would hide any overhead and task granularity problems we encountered. We decided to divert from the parallelization effort to investigate using fixed-point arithmetic.

Fortunately, most of the data was represented as indices into a table of probabilities for storage efficiency. This table contained 256 entries, ranging between zero and unity, quantized logarithmically. Therefore, the indices themselves were scaled log probabilities. Multiplication of probabilities in the original program could, of course, easily be converted to addition of corresponding log probabilities. The Viterbi algorithm also finds the maximum of the path probabilities to a node, which is equivalent in both log and linear domains. We converted the program to use log probabilities, and obtained the same results as before.

The next phase of the conversion, changing floating-point to fixed-point, was straight-forward and quickly completed, but the execution time remained disappointingly long. Butterfly measurement tools allowed us to discover where most of the time was being spent. To speed up access to array data, we replaced array subscripting with pointer arithmetic and dereferencing. We also simplified the calling sequence by defining many of the arguments globally. These modifications caused the execution time to drop to about two minutes for a 3.5 second utterance, which seemed low enough to obtain reasonable parallelization measurements.

UNIFORM SYSTEM

The Butterfly architecture provides a very uniform environment. The processors are identical, the connections between all pairs of processors are the same, and each processor's access to all locations is fairly uniform. Although a processor can access the memory that resides on its own Processor Node somewhat faster than it can cross the switch to access memory on another node, the difference in access times is not usually significant. Equal access to all memory results in uniform interprocessor communication connections, because processors only communicate through the memory.

The Uniform System was developed to exploit the architecture of the Butterfly Parallel Processor. It is a programming methodology supported by a library of high-level functions [4]. Its goal is to simplify the problem of load balancing for the memory as well as for the processors. Balancing the load on memory is

accomplished by scattering the data evenly across the different physical memories in the machine, under the assumption that this will also spread the accesses fairly evenly, reducing the inefficiency that results when many processors attempt to access the same memory simultaneously. The load on the processors is balanced when all processors are equally busy and no processor is waiting for another to finish.

The Uniform System contains functions for scattering data structures throughout memory. One of these is AllocateScatterMatrix, which allocates and scatters a two dimensional array, or matrix, so that different rows reside on different memories. If each processor uses a different row, there is no competition for any of the memories. The Uniform System also provides for processor private memories, as well as the globally shared memory. This private memory is available to the processor for data that the processor will access often, and is located on the same board as the processor. This memory assignment reduces traffic through the switch and therefore reduces the possibility of switch contention. Functions that perform block transfers between this local memory and the shared memory are included in the Uniform System, as are other functions for allocating storage in the shared memory.

The philosophy behind the Uniform System processor management methodology views the processors as a uniform pool of workers, each of which is capable of executing any application task. Using this methodology, the programmer is only required to supply code for the tasks that operates correctly when executed simultaneously by multiple processors. This is usually easier than writing and synchronizing many different sub-programs to run each on individual processors. A program is copied to each of the processors. In most cases, the program will begin with a section of serial code that is executed on a single processor. To begin executing a section of code on multiple processors, a FOR loop, for example, the programmer can use a "task generator" to replace the FOR statement and a "worker routine" to replace the body of the FOR loop. In this case, the task generator would be GenOnIndex, which is used to apply the worker routine for each value of the FOR loop index. The task generator makes a task descriptor available to all processors, which use it, as they become free, to generate calls to the worker routine. The task

descriptor for GenOnIndex is very simple, consisting of the name of the worker routine, the range for loop index, and a mechanism for determining the next iteration of the routine to perform. Processors, using this descriptor, execute the routine repeatedly for different index values, until the index has run the range. When all processors have finished, the program, once again serial, continues executing on a single processor.

The programmer must supply a worker routine to perform one instance of the task to be performed in parallel. He must also provide a task generator of determining the next instance of the task to begin execution. The Uniform System Library includes generators such as GenOnIndex, for many common forms of program structures. A simple program might contain only a single task generator, while a more complex program might contain many, possible nested, generators.

We decided to use the Uniform System for several reasons. First, the speech recognition algorithm is essentially a single task, executed many times. This fits the Uniform System paradigm very well. Second, being novices, we were attracted by the simplicity of use of the Uniform System. Third, there are functions available in the Uniform System Library that allow automatic timing of the same program run on various numbers of processors, and this provided an easy way of evaluating the performance of the parallel implementation. Finally, because the same program can be run on one or many processors, we believed that debugging the parallel implementation would be simplified.

PARALLEL IMPLEMENTATION AND RESULTS

In our speech recognition system, both the training and spectral analysis tasks are performed "off-line" and the results stored as described in Section 3. The recognition program itself begins by reading in the word models for a speaker. Then, for each utterance, the spectral parameters for the utterance are read and stored. At this point in the program, the actual recognition task begins. This recognition task was the only portion considered for parallel implementation. Our execution time measurements began here and continued until the input utterance had been recognized, that is, a theory for the complete utterance had been obtained.

The pertinent portion of the speech recognition program can be abstracted as follows.

a. FOR all frames
b.   initialize frame
c.   FOR all words
d.   initialize word
e.   FOR all states
f.   compute state score
g.   IF (new max score)
h.   replace max score
i.   IF (new max terminal)
j.   replace max terminal
k.   determine normalization
l.   FOR all words
m.   propagate scores
n.   determine theory

The first parallel version combined lines d) through f) and parts of g) and h) into a single task and used the generator GenOnIndex, which includes a prologue task and an epilogue task in addition to the main task. The prologue task is executed only once by each processor before that processor executes the main task for the first time. In this version, the prologue included line b). Similarly, the epilogue task is executed once by each processor after all main tasks have been completed by that processor. For this program, the central task determined the maximum state score and the maximum terminal score seen by each processor. The epilogue task compared these local maxima against global maxima, replacing the global maxima if necessary. The remainder of the program, including the second FOR loop (line k) was executed sequentially, on a single processor. Figure 3-a shows the execution times for various numbers of processors for lines a) though n). Figure 3-b shows the effective number of processors vs. the actual number of processors used (execution time for one processor divided by execution time for P processors). Approximately 9 seconds was spent in the sequential portion of the program when it was run for a 3.5 second utterance. When run on 15 processors, this resulted in less than 50% utilization of the processors.

The next step was to attempt to reduce the

sequential portion of the program. We noticed that the second FOR loop (lines k and l), which propagates words scores for terminal states in the current frame to initial state scores for the next frame, could be incorporated into the first FOR loop, effectively changing the program to.

a. FOR all frames

b.   initialize frame

c.   FOR all words

d.   propagate terminal scores from previous frame

e.   initialize word

f.   FOR all states

g.   compute state score

h.   IF (new max score)

i.   replace max score

j.   IF (new max terminal)

k.   replace terminal

l.   determine normalization

m. determine theory.

This revision substantially reduced the time spent executing serial code. The execution times and processor utilization for this version of the program are shown in figure 4. For 15 processors, the execution time dropped from 15 seconds to 11 seconds for a 3.5 second utterance, and the effective number of processors rose from 6.9 to 11.2, or approximately 75% utilization.

CONCLUSIONS AND FUTURE RESEARCH

Our work on this project has shown that the Butterfly architecture is suitable for continuous speech word recognition. The decomposition of the algorithm into tasks that match one word to one frame of input speech provided a granularity that made efficient use of the processors. The memory and processor management functions of the Uniform System made parallelization of the algorithm surprisingly easy and rapid.

In the near future, we hope to extend the current research to include a grammar and larger vocabulary tasks. The grammar will require search of a space that is much too large to search exhaustively. The search will have to be pruned, thus presenting a more challenging parallel implementation task.

REFERENCES

1. "Butterfly Parallel Processor Overview", BBN Laboratories Incorporated, December 18, 1985, pp 9-14.

2. Y.-L. Chow, R. Schwartz, S. Roucos, O. Kimball, P. Price, F. Kubala, M. Dunham, M. Krasner, J. Makhoul, "The Role of Word Dependent Coarticulatory Effects in a Phoneme-Based Speech Recognition System", IEEE Int. Conf. Acoust., Speech, Signal Processing, Tokyo, Japan, April 1988.

3. L.R. Bahl, F. Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", IEEE Trans. Pattern Analysis Machine Intelligence, Vol2. PAMl-5, No. 2, March 1983.

4. "The Uniform System Approach to Programming the Butterfly Parallel Processor (Draft)", BBN Laboratories Incorporated, December 1985,

FIGURES
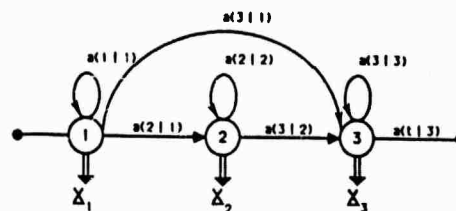
Fig. 1. Phoneme Model
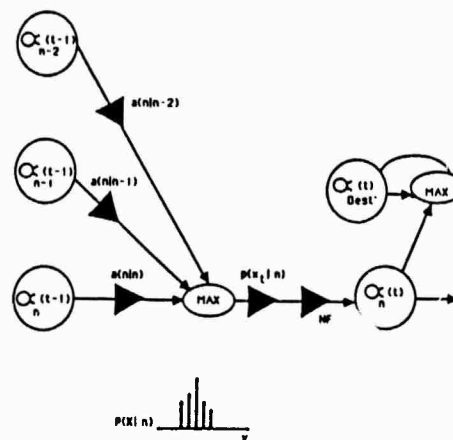
Fig. 2. State Score Computation
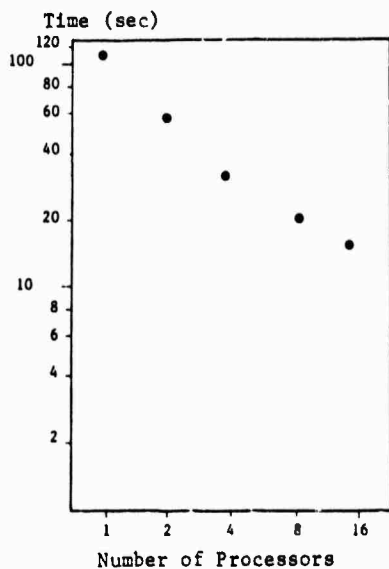
FIG. 3A
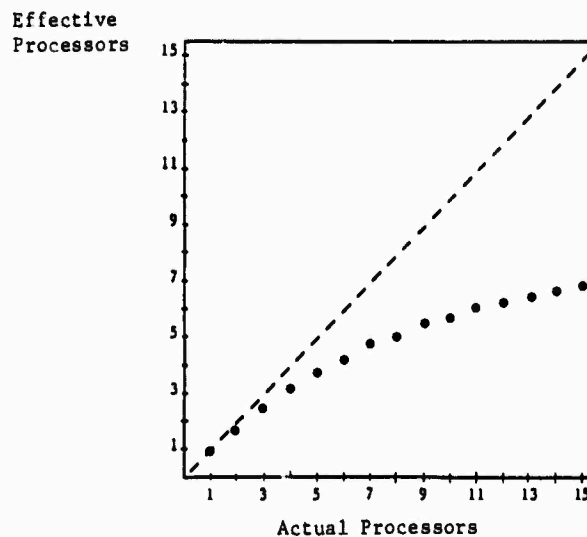
FIG. 3B

FIG. 3   INITIAL PARALLEL IMPLEMENTATION

    A.   Execution time in seconds for a speech utterance of 3.5 seconds.

    B.   Processor Utilization, The dashed line represents 100% utilization.
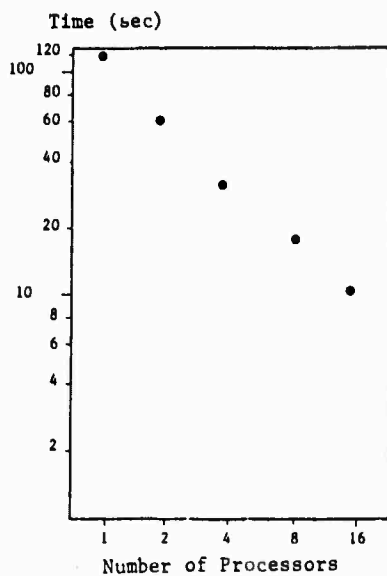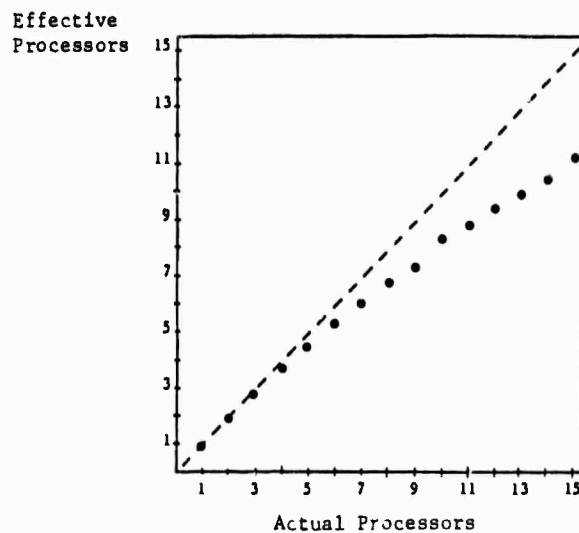

FIG. 4A

FIG. 4B

FIG. 4   FIRST REFINEMENT OF PARALLEL IMPLEMENATION

    A.   Execution time in seconds for same utterance.

    B.   Processor utilization.

A BENCHMARK FOR SPEAKER-DEPENDENT RECOGNITION USING THE TEXAS INSTRUMENTS 20 WORD AND ALPHA-SET SPEECH DATABASE

David S. Pallett

Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

ABSTRACT

This paper presents the results of performance assessment tests conducted on one commercially available speaker-dependent template-matching speech recognizer, using a widely available speech database. Test vocabularies include the Texas Instruments 20 word test vocabulary and the 26 letters of the spoken English alphabet (the alpha-set). For the 20 word set, overall recognition accuracy was 99.24%, and for the alpha-set it was 84.88%. Comparisons are made with the performance of research systems which use both template matching and feature-based technologies, as well as with the results of tests on commercially available recognizers of 5-7 years age. The intended purpose of these measurements is to provide a benchmark for comparing the results of tests of more sophisticated systems.

INTRODUCTION

As the performance of speech recognition technology improves, more challenging test material is required in order to demonstrate the capabilities of improved systems. For speaker-dependent isolated word recognition, widespread dissemination and use of the 20 word Texas Instruments (TI) speech database (first used in Doddington and Schalk's study of the state-of-the-art in 1981 [1]), has provided a valuable research resource and measures of performance that serve as benchmarks for this 20 word vocabulary. However, as performance of the technology has improved, the value of this database has declined because it may no longer provide substantial challenge to the current state-of-the art. More challenging speech test vocabularies and databases are required in order to demonstrate improved capabilities.

In large vocabulary natural language systems, the spoken letters of the English alphabet, or the "alpha-set", may be widely used to introduce the spelling of new words in the lexicon. This application has been termed "spellmode". In such an application, the use of syntax to restrict the vocabulary is obviously inappropriate, and the required use of special-purpose alphabets such as the International Civil Aviation Phonetic Alphabet is probably undesirable. The use of the alpha-set is natural in such an application.

At the time that the TI 20 word database was collected, the same talkers also provided tokens for the alpha-set [2], and this speech database is now in the public domain. The availability of this test material provides a means for comparative tests on both the 20 word database and the alpha-set for the same set of talkers, and increases the value of the original 20 word database by providing more challenging material from the same group of test talkers that was obtained under identical environmental conditions.

This paper presents preliminary results on tests of performance on the TI 20 word vocabulary and the alpha-set for a representative commercially available speaker-dependent recognizer costing approximately $1000. These data are intended to provide benchmarks of performance for comparison of the performance of more sophisticated recognition algorithms, using speech database material that is widely available. More detailed analysis of this data is being conducted and at least two other commercially available recognizers are to be studied.

TEST PROCEDURES

The tests reported upon in this paper were conducted using procedures outlined in a recent paper [3]. They reflect suggestions on experimental design, data analysis and documentation from the IEEE Speech I/O Technology Performance Evaluation Working Group. Material included in this section follows the format suggested in this reference.

Experimental Design

These tests were intended for benchmark purposes. The TI 20 word vocabulary and the alphabet were used in separate tests, with no use of syntax to control the active recognition vocabulary. The use of these vocabularies may be representative of an application such as "spellmode", but no explicit effort is taken to model an application.

Test Talker Population

Eight males and eight females comprise the test talker population, with no effort taken to control dialect.

Test Vocabulary

The 20 word vocabulary consists of the words "yes, no, erase, rubout, repeat, go, enter, help, stop, start" and the digits "zero" through "nine". The alpha-set consists of the letters "a" through "z". All words were spoken as discrete utterances. It is interesting to note that the available tokens in the database could be recombined to yield an "alphadigit" set as used in other studies [4,5], but this study sought to direct attention to a comparison of performance for the 20 word and alpha-sets.

Training

The database includes 10 tokens of each of the 46 words for each talker. These tokens are intended for use in training or enrollment. This material was used for enrollment in accord with the manufacturer's recommendations. Typically, the first token was used for 'enrollment', and three additional tokens were used to 'update' the resulting reference patterns or templates. Training was implemented automatically, and no attempt was taken to optimize the reference template set.

Environment

Test material was obtained in a quiet sound-isolation booth with a cardiod dynamic microphone placed approximately 2 inches from the talker's mouth. The speech signal-to-noise ratio is believed to exceed 40 dB, but (to date) has not been measured.

Recorded Test Material

The speech signal was initially digitized with a 12-bit A/D converter at a 12.5 kHz sampling rate. The digital data were made available to the National Bureau of Standards by Texas Instruments for use in the public domain. An analog signal was reconstructed using a D/A converter, using a 6.3 kHz antialiasing filter. This audio signal was then recorded using commercially available PCM/VCR technology with a digital mastering processor and a video cassette recorder.

One audio channel on the PCM/VCR recorded material provides a recorded modem signal with ASCII character string data that precedes each utterance recorded on the other audio channel. The use of this format and 'header' data facilites automatic enrollment and scoring [6].

Playback of the recorded material provides two line-level audio signals, one for the modem and one with the test material. The line-level audio signal with the test material was used as input to a mixer, with the microphone level output of the mixer used as input to the recognizer. Headphones driven by the mixer were used to monitor the signal as desired.

Calibration tones provided on the PCM/VCR recorded material were used to establish system gains, and tests were conducted using the recognizer manufacturer's routines to establish appropriate recognizer gains. Once gains were established, they were fixed, and no effort was taken to optimize gains for improved performance.

Statistical Considerations

There are a total of 5120 test tokens for the 20 word vocabulary (16 test tokens for each of the 20 words for each of the 16 talkers). There are a total of 6655 valid test tokens for the alpha-set. One test token of one letter ("s") for one talker (f5) has been found to contain only breath noise. For each of the 16 talkers, there are 10 training tokens available for each of the 46 words in the two vocabularies of the database, for a total of 7360 training tokens. The total number of tokens in the database is thus 19135 tokens.

Since the total number of errors per talker is small, the precision associated with these data is unknown.

Several repetitions of tests for individual talkers were conducted in order to assess the variability between repeated tests. These tests included repeated enrollment and repeated use of the of the test material on a given template set. In general, there has been very good repeatability, typically varying in one count of the total number of substitutions. Although the number of errors per talker is larger for the alpha-set, the variability in the count of the total number of

substitutions is typically three or four.

BENCHMARK DATA

20 Word Vocabulary

Overall Scores for 5120 tokens
for 8 males and 8 females

Correct Recognition Percent: 99.24% (5081)
Substitution Percent: 0.61% (31)
Deletion Percent: 0.06% (3)
(No Insertions)
Rejection Percent: 0.10% (5)
Ratio of total errors
     to rejections: 6.8

Figure One indicates the distribution of responses for the 20 word vocabulary. In this matrix representation, the input words are listed along the rows, and the recognizer's responses are shown in the appropriate column.

Alpha-set

Overall scores for 6655 valid tokens
for 8 females and 8 males

Correct Recognition Percent: 84.88% (5649)
Substitution Error Rate: 14.92% (993)
Deletion Percent: 0.03% (2)
(No Insertions)
Rejection Percent: 0.17% (11)
Ratio of total errors
     to rejections: 90.4

Figure Two indicates the confusion matrix for the alpha-set.

The intended test procedure was to disable the reject capability of the recognizers under test to facilitate comparisons. For this recognizer, it was not possible to do so. Following the manufacturer's recommendation, the acceptance threshold was set to its maximum value, and no restrictions were imposed on the 'closeness' of best and next-best scores. This results in a very low, but non-zero rejection percent. Performance on the alpha-set might be improved by the imposition of appropriate reject criteria.

For the 20 word set, there are 5 rejections for the male talkers and no deletions, and no rejections and 3 deletions for the female talkers. For the males, recognition accuracy and substitution error percents are, respectively, 98.94% and 0.86%, with corresponding data for the females 99.53% and 0.35%.

For the alpha-set, there are 10 rejections for the male talkers and no deletions, and 1 rejection and 2 deletions for the female talkers. For the males, recognition accuracy and substitution error percents are, respectively, 83.4% and 16.3%, with corresponding data for the females 84.9% and 14.9%.

DISCUSSION

In comparing error rates for the two test vocabularies, the substitution error rate is approximately 20 times larger for the alpha-set, reflecting the greater difficulty of recognizing a vocabul v consisting exclusively of monosyllables (with the sole exception of "W"), containing several highly confusable subsets, and with a branching factor that is 30% larger.

The small number of substitution errors observed for the 20 word vocabulary (31) is believed to fairly represent the state-of-the-art of currently available low-cost recognizers. Further data are to be obtained on other recognizers, including the use of other approaches including stochastic modelling. By comparison with the results of Doddington and Schalk's 1981 benchmark data, the error rate is half that of the second-best recognizer in their tests (a template matching unit then having a nominal price of $65,000).

As previously noted for the TI 20 word data base [1], there is considerable variation between individual talkers' scores. For the 20 word set, individual scores range from 97.5% to 100%, while for the alpha-set the range is from 74.3% to 91.8%. In general, "sheep" and "goats" retain their relative rank-order places when comparing results for the two vocabularies. These variations underscore the need for adequate population sampling and large enough test data bases for statistical validity.

In studying the confusion matrix that results from separate consideration of members of the E-set as input (Figure Three), it is evident that the bulk of the substitution errors occur for the E-set (the letters "B,C,D,E,G,P,T,V" and 'Z'). There are a total of 2304 test tokens in this subset. There are a total of 1546 correct responses and 754 substitution errors, for a subset recognition accuracy of 67.1% and a substitution percent of 32.7%. The 754 substitution errors for the 9 word E-set comprise approximately 76% of all substitution errors for the 20 word Alpha-set. Approximately 98% of all substitution errors for E-set input tokens

69

fall within the E-set.

With.n the E-set, overall recognition accuracies for individual letters range from 92.9% for "E" to 53.1% for "D", with significant variations occurring for different talkers.

Previous measures of the ability of template matching systems to perform fine phonetic distinctions as cited in Cole et al. [7] indicate recognition accuracy for the E-set at about 60%. The present measurements on a commercial product suggest slightly better performance, with considerably better performance for "E" (92.9%) and "G" (90..) than for other members of the subset s h as "B" (58.6%) and "D" (53.1%).

In Cole's work comparing template matching and feature based recognition, an alpha-set data base of 2080 tokens was used (4 tokens of each letter produced by 10 female and 10 male talkers). The system under study was operated in a speaker-independent mode, with a procedure used to ensure that the test talker's data were consistently deleted from the training material. Without tuning (adaptation to individual talker's speech), an overall error rate for the alpha-set of 10.5% was obtained, in contrast with the error rate of 14.92% found for the speaker-dependent recognizer in this study.

For the E-set, Cole cites an error rate of 14% in contrast with the 32.7% in this study. Using tuning (on the limited number of tokens available for each letter for each speaker in his data base) and improved algorithms, Cole indicates that an error rate of 6% was obtained, approximately one-fifth that of this commercially available template-matching recognizer. This comparison suggests the strength of the speaker-independent feature-based recognition technology when compared with current technology. Further comparisons with the performance of speaker-dependent (or adaptive) systems using stochastic models should be informative.

SUMMARY

This paper reports on preliminary tests conducted using a widely available speech data base in the public domain and a commercially available recognizer using template matching technology. For the 20 word vocabulary used in these tests, a recognition accuracy of 99.24% was measured, while for the spoken English alphabet, recognition accuracy was 84.88% The 9 members of the E-set are responsible for 75.9% of all substitution errors for the 26 letters of the spoken alphabet.

These tests suggest that the performance of current low-cost commercial products using template matching technology is slightly superior to results reported for research systems of 5 to 7 years ago and to that of commercially available systems coating as much as $65,000 of that era. The tests also suggest that performance is inferior to more sophisticated systems using stochastic modelling and/or acoustic-phonetic feature based recognition.

REFERENCES

[1] G.R.Doddington and T.B.Schalk, "Speech Recognition: turning theory to practice", IEEE Spectrum, September 1981, pp.26-31.

[2] T.B.Schalk, "The Design and Use of Speech Rec gnition Data Bases", Proceedings of the Workshop on Stardardization for Speech I/O Technology, D.S.Pallett, (ed.), National Bureau of Standards, March 1982.

[3] D.S.Pallett, "Performance Assessment of Automatic Speech Recognizers", Journal of Research of the National Bureau of Standards, Vol. 90, No. 5, September-October 1985, pp.371-387.

[4] N.R.Dixon and H.F.Silverman, "What are the significant variables in dynamic programming for discrete utterance recognition?", Proceedings of ICASSP'81 (Atlanta), pp.728-731.

[5] L.F.Lamel and V.W.Zue, "Performance Improvement in a Dynamic-Programming-Based Isolated Word Recognition System for the Alpha-Digit Task", Proceedings of ICASSP'82 (Paris), pp.558-561.

[6] D.S.Pallett, "A PCM/VCR Speech Database Exchange Format", to appear in Proceedings of ICASSP'86 (Tokyo).

[7] R.A.Cole, R.M.Stern and M.J.Lasry, "Performing Fine Phonetic Distinctions: Templates vs. Features", in Conference Record of "Toward Robustness in Speech Recognition", W.A.Lea, (ed.), Santa Barbara, CA November, 1983.

Figure One — Confusion matrix (20 word vocabulary)

	yes	no	erase	rubout	repeat	go	enter	help	stop	start	one	two	three	four	five	six	seven	eight	nine	zero
yes	255																			
no		253				e														
erase			256																	
rubout				256																
repeat					246														9	
go		4				252														
enter	2						255													
help								256												
stop									256											
start										256										
one											254				1					
two												256								
three													256							
four														256						
five								3							253					
six																253				
seven																	255			
eight																		255		
nine								1											253	
zero		2			1															253

Figure One: Confusion matrix representing responses for the 20 word vocabulary.

Figure Two — Confusion matrix (alpha-set)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	223	1						1	1	20						1			3							
b		150	2	30	31										6				1		25					5
c		14	196	2	6		2								11				6		4					5
d		73	4	136	33	4									9				18		5					4
e		11		1	238		2								6				2							
f					1	207													48							
g		6	6	4			232		1						1				6							
h	1		2	1				254																		1
i							1		254															1		
j	10									243	2															1
k	28								7		221															
l					7	1						248														
m													233	20												
n													5	248												
o															256											
p		21	4	22	19	1										154			32		3					
q																	236		20							
r																		255								
s					22	1													228			3				
t		9	4	18	23		5								23	3		161		141						
u															14				241							
v	75	3	9	13					2						3				138					13		
w					1														1	153						
x									1									2		151						
y							1															255				
z		29	4	18	18		1		1						3				37						141

Figure Two: Confusion matrix representing responses for the alpha-set.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b		150	2	36	31											6				1		25				5
c		14	196	2	6		2									11				6		4				5
d		43	4	136	33	4										9				18		5				4
e		1		4	238		2									6				2						
g		6	6	4			232		1							1				6						
p		21	4	22	19	1										154				32	3					
t		9	14	18	23		5									23	3			161						
v		75	3	9	13					2								3				138				13
z		29	16	18	18		1		1							3				37						141

Figure Three: Confusion matrix representing responses for the E-set.

SECTION 2

REVIEW OF NEW GENERATION SYSTEM FOR
ROBUST SPEECH RECOGNITION

ROBUST SPEECH RECOGNITION: INITIAL RESULTS AND PROGRESS

Periagaram K. Rajasekaran
and
George R. Doddington
Texas Instruments Inc.
Computer Sciences Center
P.O. Box 226015, MS 238
Dallas, Texas 75266, USA
Tel. (214) 995-0389

ABSTRACT

This paper describes the initial efforts and results in the pursuit of robust speech recognition for military applications. The initial goal of this prog is 98% correct recognition of 100 words der noisy and stressful conditions in a speaker-dependent recognition mode. The next goal is a similar performance with 200 connected words, initially with speaker-dependent recognition with progress towards speaker independence. The program is being executed along three interacting dimensions of data base, algorithm development and implementation. Progress achieved along each dimension promises the viability of robust recognition within certain limitations.

1. INTRODUCTION

This paper describes the initial efforts and results in the pursuit of robust speech recognition for military applications. The initial goal of this program is 98% correct recognition of 100 words under noisy and stressful conditions in a speaker-dependent recognition mode. The next goal is a similar performance with 200 connected words, initially with speaker-dependent recognition with progress towards speaker independence.

Speech recognition is an empirical science and derives its power from experimentation with large amounts of speech data that reflect the conditions of operation. This mandates the establishment of one or more data bases that will help develop robust recognition. Section 2 briefly describes the various data bases that are available for this purpose. Section 3 describes algorithms that have been developed and used by Texas Instruments and believed to be the initial set from which the ultimate robust recognition system will be developed. Section 4 presents the various recognition experiments that have been conducted and the results obtained. Implementation aspects form the subject of Section 5.

2. DATA BASES

Texas Instruments (TI) has collected a set of three data bases under the auspices of the Darpa Robust Speech Recognition program and other TI programs. A fourth data base, which is the main data base to be used, was defined by TI, Armstrong Aerospace Medical Research Laboratory (AAMRL) of Wright-Patterson Air Force Base (WPAFB) and Lincoln Laboratory (LL), and is being collected by AAMRL. The four data bases are:

1. Simulated Stress Data Base
2. Advanced Fighter Technology Integration (AFTI) Connected Word Data Base
3. LHX Vibration Data Base
4. Robust Recognition Data Base

2.1 Simulated Stress Data Base

Establishing this data base was motivated by the need to fuel the initial algorithm development and experimentation efforts for robust recognition. Psychological and physiological stress on a speaker manifest themselves as variabilities in the acoustic signal produced. Typical of the variabilities are the changes in the spectral slope, fundamental frequency, formant locations, level and duration of the acoustic events of the speech signal [1]. Stress-like degradations of the speech signal were elicited by asking the speaker to produce speech with vocal efforts/effects corresponding to Normal, Fast, Loud, Shout, and Soft conditions as well as with Noise Exposure (95 dB) in the ears. The vocabulary consisted of 105 word including monosyllabic, polysyllabic and confusable words such as "one", "destination", "advisory", "six", "sixty", "fix" etc. Training data consisted of 5 samples of each of the 105 words in a random order under normal conditions, and test data consisted of 2 samples of each word under each stress condition listed above. Data were collected from 5 adult male and 3 adult female speakers, and digitized at a sampling rate of 20 kHz using a 16-bit A/D converter. The data used in our

experiments were downsampled to 8 kHz from 20 kHz by means of a downsampling program.

2.2 AFTI Connected Word Data Base

This data base was established to support the AFTI F-16 voice recognition efforts in the government electronics group at TI. The subject was asked to wear typical F-16 helmet and an oxygen mask with an embedded M 101 microphone. A nominal level of F-16 spectrum noise (65 dB spl) or a higher level (80 db spl) was played at the subject's ears along with his own voice feedback. The utterances to be said were prompted on a screen with the subject in a quiet sound booth. Data were digitized at 20 kHz and downsampled to 8 kHz. The speaker was subjected to a variety of conditions to elicit the effects of stress and noise. An initial set of 87 isolated words and about 230 phrases, generated by an application-specific finite state grammar, were collected under normal conditions for enrollment purposes. The test data conditions were: (1) Normal with nominal noise, (2) Normal with the higher level of noise, (3) Normal with no noise, (4) Fast mode of speech, (5) Loud level of speech, (6) Soft level of speech, (7)Deliberate manner of speech (clear enunciation), (8) Twist, where the speaker had to turn his head between 90 and 180 degrees to look at the prompt, and (9) Back, where the subject was lying on his back while uttering the prompts. Conditions 4 through 9 also used nominal noise level at the subject's ears. The test data consisted of 153 phrases, generated by the application grammar, and the 87 isolated words. The vocabulary contained typical command control words used in the AFTI program.

2.3 LHX Vibration Data Base

Vibration conditions could alter the acoustic characteristic of a speaker and thereby degrade the performance of speech recognizers. This data base was established by TI's government electronics group to evaluate the application potential of TI speech recognizer for the LHX helicopter. Two sets of vocabulary, one monosyllabic and another polysyllabic, were chosen. There were 50 isolated words in each set. The subject was seated in a helicopter seat mounted on a vibration platform and wore a helmet with army issue M87 noise-cancelling microphone. A group of four male speakers uttered the first vocabulary set, and another group of four male speakers uttered the second set. The vibration conditions were established by choosing a spectra and amplitudes corresponding to the LHX vibration conditions. The most severe condition was the right turn in which the helicopter is turning against the direction of the rotor blades. There were four training sessions with one token per word, all in succession and on the same day. The speakers were not subjected to any vibration during the training sessions. The test sessions were collected the following day and the order of the condition was randomized individually for each subject. The vibration conditions simulated the following: (1) Right Turn, (2) Full Power Climb, (3) Left Turn, (4) Hover, (5) Approach, (6) Level Flight, and (7) No Vibration. Data were collected using PCM/VCR equipment, and is not available in digitized form.

Figure 1 shows the spectrograms from a male speaker saying the word "PLAN VIEW" for no vibration condition and the severe vibration condition. The waveforms clearly show the modulating effect of vibration.

2.4 Robust Recognition Data Base

This data base consists of 539 phrases (training) and 219 phrases (test) generated by a fighter aircraft language model described by a finite state grammar and 207 words. The first phase consists of ten Air Force qualified personnel as the speakers uttering both discrete and connected utterances under the following training conditions: (1) Normal with no noise or stress, (2) Lombard Effect, (3) Loud, (4) Fast, and (5) Normal with no flight gear. The test data conditions will consist of (1) Normal with no noise or stress, (2) additive noise conditions of broadband and discrete noise, (3) stress conditions due to vibration, positive pressure breathing, work load, and Lombard effect, and (4) simulated stress by varying vocal efforts. In the second phase a smaller set of speakers, qualified as hazardous-duty pilots, will be used to collect training data as in phase 1, but test data will include g-force loading effects. A third phase will establish a validation data base to test the algorithms without the bias of the data bases on which they were developed. This data base is collected using PCM/VCR data acquisition means, and will be digitized by TI for further use and distribution. Fifty percent of the phase I enrollment is complete at this time.

3. ALGORITHMS

The baseline algorithm used is called a Principal Spectral Component (PSC) algorithm in which only spectral information is used [2]. In an enhanced

system, the rms energy of a frame of speech is used as well and is called the Principal Feature Vector System [3]. A frame-pair frame-specific (FPFS) PFV system comprehends the nonstationarity as well as spectral dynamics of speech signals. This is described in [4]. Connected word recognition [CWR] is a high level construct that supplements the isolated word recognition to perform a truly connected word recognition by pruning sentence hypotheses according to a grammatical structure.

3.1 Baseline Method (PSC)

Figure 2 shows the generic block diagram of the method. The LPC parameter vector characterizing a frame of speech (test or reference) is transformed to spectral amplitudes (on a dB scale) normalized to the frame energy using a simulated filter bank. A critical-band filter bank [5] was used in the study. The filter bank amplitudes constitute a vector that may be characterized as normally distributed with mean vector depending on the word(hypothesis), and a covariance matrix. This covariance mat. x may be estimated by pooling all available data for the entire vocabulary. Implicit in this process is the assumption that all frames are statistically independent and have the same covariance matrix. A reference template, then, consists of a sequence of hypothesis-dependent mean vectors of filter bank amplitudes, and its statistical variability is described by a single covariance matrix. The recognition problem is to compute, given the input characterization, the likelihoods corresponding to each word hypothesis, and choose that with the largest likelihood. This corresponds to maximum likelihood decision.

In general, the amplitudes of adjacent filters are highly correlated and provide potential for reduction of dimensionality of the feature vector. The filter bank amplitudes are rotated by the eigenvectors of the covariance matrix so that the resulting transformed features are statistically uncorrelated [6]. These features are ranked in decreasing order of statistical variance (eigenvalues), and the least significant features are discarded resulting in a dimensionality reduction. Finally each of these new features is scaled so that its variance is unity. The resulting features are called principal spectral components(PSC), and previous studies have established correlations with perceptual space for certain classes of sounds [7]. A Euclidean distance in this feature space is used as the metric to compare input and reference frames of speech data.

3.2 Enhanced Method (PFV)

The energy-time profile of a speech signal appears to be rich in information for human recognition. It is only reasonable to include the rms energy of a frame of speech signal as an additional feature to the filter bank spectral amplitudes of the the PSC method. The enhanced set can again be orthogonalized statistically as in PSC method, and the higher variance components chosen. The resulting vector is called the Principal Feature Vector (PFV). The Euclidean distance metric and the statistical optimality of maximum likelihood decision is maintained.

3.3 Frame-Pair Frame-Specific Method (FPFS)

In the PSC and PFV methods, as in most isolated word recognition algorithms, the feature vectors of successive reference frames of a word template are assumed to be statistically independent. Further, they are assumed to be identically distributed except for their mean vaue. This is generally not true for speech signals. Adjacent frames are not spectrally (or acoustically) independent. Further it is only reasonable to believe that the covariances of the feature vectors depend on the acoustic event (i.e. the reference frame under consideration) thereby allowing a nonstationary model. The effct of spectral dynamics is brought out by considering a frame-pair vector made up of concatenating feature vectors from adjacent reference frames. The concept of principal component analysis can then be applied resulting in a FPFS PFV system. This method has been successfully applied to speaker-independent recognition of digits [4].

3.4 Connected Word Recognition (CWR)

Figure 3 shows a block diagram of the CWR system. An isolated recognizer outputs all the words that are hypothesized along with corresponding distance scores and durations. These form inputs to the sentence recognizer which invokes a grammar to compute distance scores for all legal sentence hypotheses. The distance measure for the sentence has three parts: the first component is the sum of individual word distance scores multiplied by corresponding word durations, the second is a penalty for overlap or underlap of adjacent words, and the third is a silence (null speech) distance measure. An important feature of the recognizer is that the sentence hypothesizer does not control the isolated

word recognizer by any feed back. It fits all possible sentences from the word hypotheses and their time marks, and invokes the grammatical constraints to search only the legal paths to output a recognized sentence.

4. EXPERIMENTS AND RESULTS

In all of the experiments, the front end signal processing is a 10-th order autocorrelation method of LPC characterization at a frame period of 20 ms with a window length (Hamming windowed) of 30 ms. The sampling frequency is 8kHz. Except in the case of experiments with the vibration data base, 16-bit data was used and the results are for experiments conducted on a VAX computer. In the case of vibration data, a TIPC speech recognition system with a codec (equivalent to 13-bit dynamic range A/D converter) operating nominally at 8kHz sampling frequency was used. For all of the experiments, TRAINING WAS DONE UNDER NORMAL CONDITIONS, which are different from the test conditions. This is deemed important because of the general inability and inconvenience of training under the stress conditions. It is only believed that training under operating conditions, if possible, will impove performance.

4.1 Experiments with Simulated Stress Data Base

The baseline system (PSC), enhanced system (PFV), and the FPFS PFV were used for recognition. PSC algorithm used a covariance matrix that was derived from an entirely different data base. PFV algorithm used a covariance matrix from all the simulated stress conditions pooled over all speakers and words. In the FPFS PFV system, the covariances were obtained in a reference frame-pair frame-specific manner for each word over all conditions from all speakers except the speaker being tested.

The substituition rates are shown in Table 1 for the various cases. It is easily seen that the basic system that performs well under normal conditions degrades rather rapidly under the simulated stress conditions. The addition of rms energy to the features reducess the average substitution rate by about 30%. Experiments performed with 10 ms frame period characterizations have yielded additional reduction of about 15% but at the cost of quadrupling the computations.

The much expected improved performance from FPFS system did not materialize, presumably due to the poorer estimates of the covariance matrices due to inadequate data. Further analysis of the results and methods to improve the performance are under investigation.

4.2 Experiments with AFTI Connected Word Data Base

In CWR, an initial enrollment is done on isolated words to create isolated word templates. These are used to segment connected word phrases with the guidance of the task-grammar to perform enrollment in the connected word mode. These "connected" enrollment templates are updated with additional training phrases and grammar control to obtain stable templates of the words in the connected speech context. The word hypothesizer used in this experiment was a PFV system. The experiment has been completed with only one speaker's data. The results are shown in Table 2. Notice that the fast connected speech is very poorly recognized. This was not surprising for two reasons: (i) the normal connected speech is already rapid, and (ii) this particular speaker misarticulated many syllables in fast connected speech. The results are very promising otherwise, and the importance of this must be underscored by the fact that connected utterance modality is more natural than isolated word modality. Any improvements in isolated word hypothesizing will result in increased benefits in the connected word mode.

4.3 Experiments with LHX Vibration Data Base

PFV system as implemented on a TMS 32010 signal processor in a TIPC speech recognition system was tested with the vibration data. This obviated the need for digitizing the sizable amount of data, and provided means for quickly establishing performance under one of the stressful conditions, namely vibration. A study of the tradeoff between substitution and rejection was performed for both monosyllabic and polysyllabic vocabularies. These are shown in Figures 4 and 5. An analysis of the results did not show any trend with the conditions leading to the conclusion that none of the vibration conditions was particulary severe for the recognizer. The poorer performance with the monosyllabic vocabulary was expected, but a significant portion of the errors came from two naive subjects whose vocal efforts varied almost by 20 dB even within a session lasting only five minutes.

5. IMPLEMENTATION

The successful robust recognition algorithm will be implemented on an advanced multi-TMS32020 processor board, called Odyssey board, with the TI Explorer, a Lisp machine, as the host. Figure 6 shows the architecural block diagram of the Odyssey board. This board is capable of 20 million multiply-accumulates per second, and will be able to handle even more computation-intensive recognition algorithms by concatenating additional boards. On-board memory of 512 kbytes will be shared by all the four processors. Data acquisition is direct through the I/O bus without the need of communicating through the host. The system can be expanded up to sixteen boards, with communication through the signal processing bus (SPB). Host communications are through the standard NuBus. CWR code for TMS32020 has been developed using a VAX simulator and is undergoing hardware/software debug. A basic operational CWR system is now forecast for mid-April 1986.

6. SUMMARY

The initial efforts and progress in the robust speech recogntition program was described. The various data bases currently available were presented along with results of the various recognition experiments conducted. Connected word recognition appears to be a valuable approach to robust recognition by harnessing the redundancy of speech input through finite-state grammar models and it will benefit significantly with improvements in isolated word recognition techniques. A powerful signal processing board under development at TI will support computationly burdensome recogntion algorithms. Investigations are underway to determine the benefits of integrating frequency measures along with spectral amplitude features. The robust recognition data base, being collected by AAMRL, will establish the limits of current technology and play catalytic role in developing more robust recognition algorithms.

7. REFERENCES

[1] Pisoni, D., R.H. Bernacki, H.C. Nusbaum, and M. Yuchtman, Some Acoustic Phonetic Correlates of Speech Produced in Noise, Proceedings of ICASSP 1985, pp. 1581 - 1584.

[2] Rajasekaran, P.K. and G.R. Doddington, Speech Recognition in the F-16 Cockpit using Principal Spectral Components, Proceedings of ICASSP 1985, pp. 882 - 885.

[3] Rajasekaran, P.K. and G.R. Doddington, Recognition of Speech under Stress and in Noise, Proceedings of ICASSP 86, April 1986.

[4] Bocchieri, E.L. and G.R. Doddington, Frame-Specific Statistical Features for Speaker-Independent Speech Recognition, Trans. ASSP, 1986 (To appear).

[5] Zwicker, E. and E. Terhardt, Analytical Expressions for Critical- band Rate and Critical Bandwidths as a Function of Frequency, J. Acoust. Soc. Am. 68(5), pp. 1523-1525, Nov. 1980.

[6] Pols, L.C.W., Real-Time Recognition of Spoken Words, IEEE Trans. comput., Vol. C-20, pp. 972-978, Sept. 1971.

[7] Pols, L.C.W.,L.J.Th.v.d. Kamp, and R. Plomp, Perceptual and Physical Space of Vowel Sounds, J. Acoust. Soc. Am., Vol. 46, pp. 458-467, Aug. 1969.
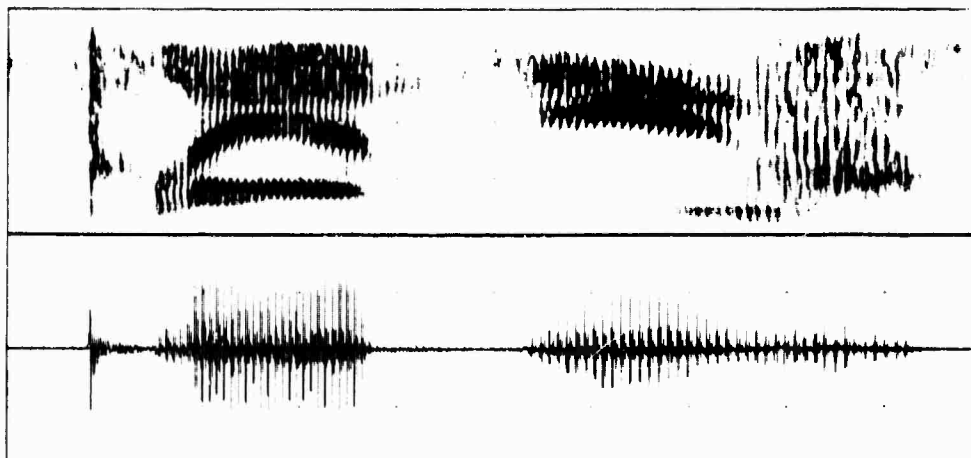
Fig. 1.a. Wideband spectrogram and waveform of the word
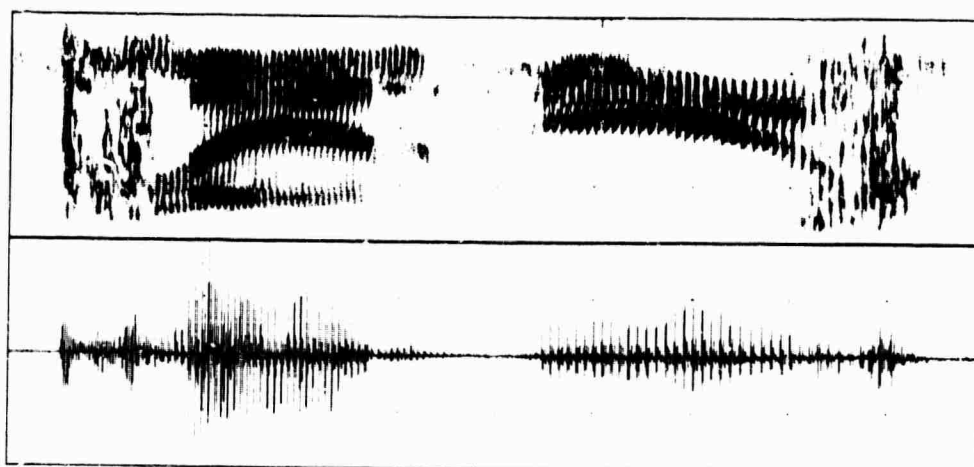"PLAN VIEW" spoken by an adult male (No vibration)

Fig. 1.b. Wideband spectrogram and waveform of the word
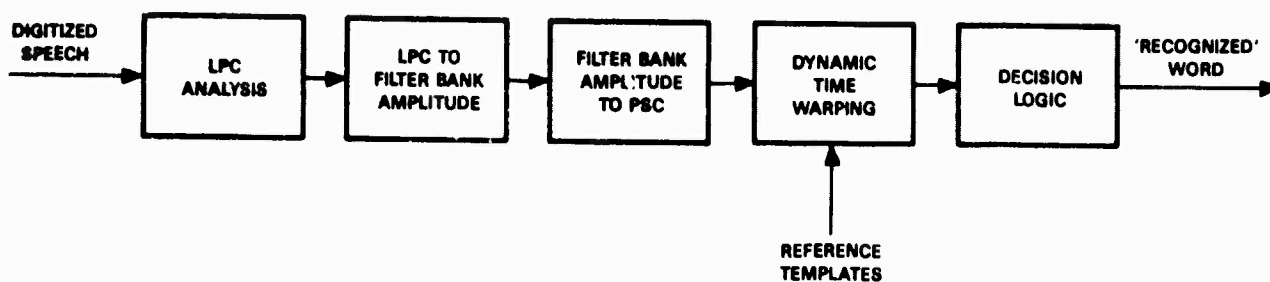"PLAN VIEW" spoken by an adult male (vibration)

Fig. 2. Block Diagram of the Principal Spectral Component
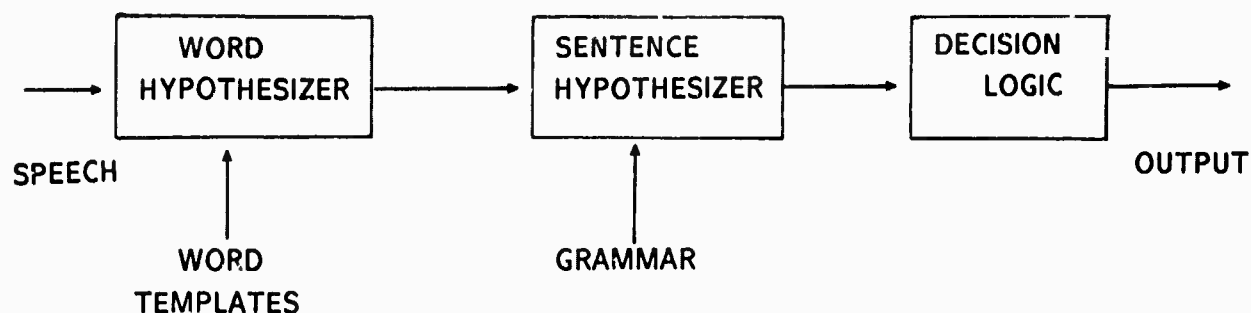Recognizer

78

```
SPEECH →  [ WORD          ] → [ SENTENCE       ] → [ DECISION ] →  OUTPUT
          [ HYPOTHESIZER   ]   [ HYPOTHESIZER   ]   [ LOGIC    ]
               ↑                    ↑
             WORD                GRAMMAR
           TEMPLATES
```

Fig. 3. Connected Word Recognizer

TABLE 1

SUBSTITUTION RATE (%) FOR SIMULATED STRESS DATA BASE EXPERIMENTS

METHOD	NORMAL	FAST	LOUD	NOISE	SOFT
PSC	1.1	10.2	24.4	13.8	11.9
PFV	0.9	8.9	19.9	9.2	4.6
FPFS PFV	3.1	17.3	16.7	10.1	10.6

TABLE 2

AFTI CONNECTED WORD RECOGNITION PERFORMANCE

153 Phrases, 1 Speaker

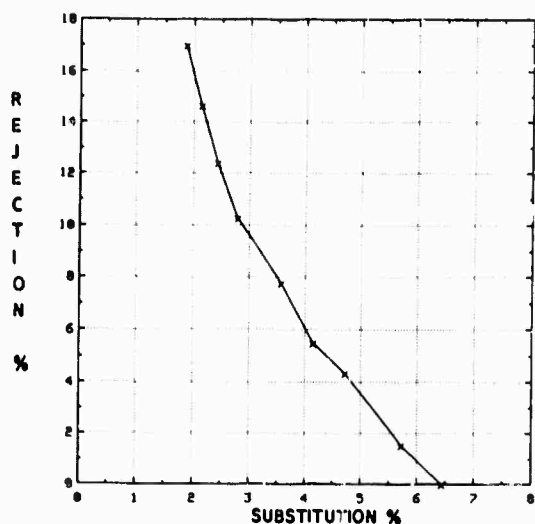| CONDITION	SENTENCE ERROR(%)		WORD ERROR(%)		
	SUB	REJ	SUB	INS	REJ
Normal	2.0	0.0	0.5	0.0	0.0
NORMAL (no noise)	3.9	1.3	1.1	0.0	0.2
Normal (loud noise)	5.2	2.6	1.1	0.2	0.3
Deliberate	2.0	0.7	0.6	0.0	0.2
Loud	6.5	2.0	1.3	0.0	0.6
Back	8.5	1.3	1.6	0.2	1.0
Twist	9.8	1.3	1.3	0.2	1.0
Soft	7.2	4.6	1.5	0.0	1.0
Fast	44.4	30.1	12.5	0.0	33.3

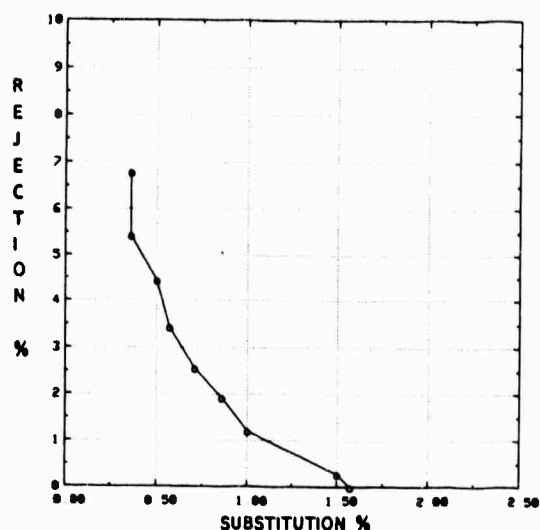Fig. 4. Recognizer Operating Characteristic for LHX Vibration Data (Monosyllabic Vocab.)

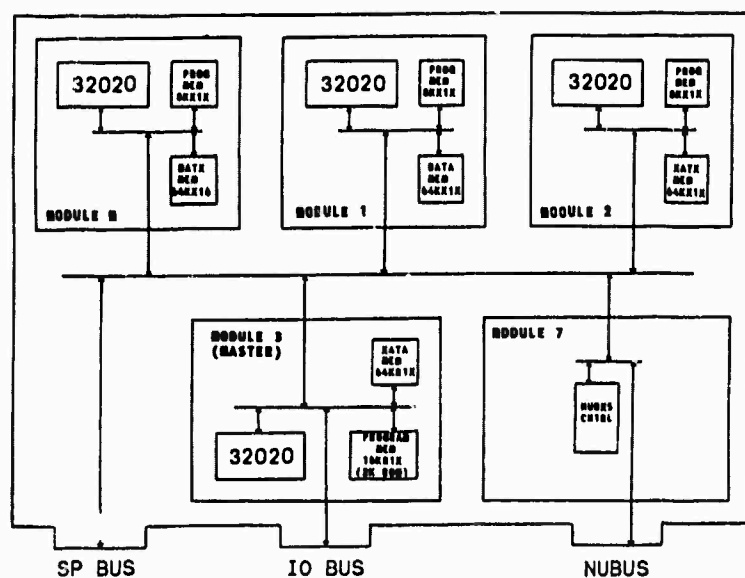Fig. 5. Recognizer Operating Characteristic for LHX Vibration Data (Polysyllabic Vocab.)

Fig. 6. Signal Processor Board block diagram

ROBUST HMM-BASED TECHNIQUES
FOR RECOGNITION OF SPEECH PRODUCED UNDER STRESS AND IN NOISE

Douglas B. Paul, Richard P. Lippmann, Yeunung Chen, Clifford J. Weinstein

Lincoln Laboratory, Massachusetts Institute of Technology
Lexington, Massachusetts 02173-0073

ABSTRACT

Substantial improvements in speech recognition performance on speech produced under stress and in noise have been achieved through the development of techniques for enhancing the robustness of a baseline isolated-word Hidden Markov Model recognizer. The baseline HMM is a continuous-observation system using mel-frequency cepstra as the observation parameters. Enhancement techniques which were developed and tested include: placing a lower limit on the estimated variances of the observations; addition of temporal difference parameters; improved duration modelling; use of fixed diagonal covariance distance functions, with variances adjusted according to perceptual considerations; cepstral domain stress compensation; and multi-style training, where the system is trained on speech spoken with a variety of talking styles. With perceptually-motivated covariance and a combination of normal (single-frame) and differrential cepstral observations, average error rates over five simulated-stress conditions were reduced from 20% (baseline) to 2.5% on a simulated-stress data base (105-word vocabulary, eight talkers, five conditions). With variance limiting, normal plus differential observations, and multi-style training, an error rate of 1.8% was achieved. Additional tests were conducted on a data base including nine talkers, eight talking styles, with speech produced under noise exposure (Lombard condition), and speech produced under two levels of motor-workload stress. Substantial reductions in error rate were demonstrated for the noise and workload conditions, when multiple talking styles, rather than only normal speech, was used in training. In experiments conducted in simulated fighter cockpit noise, it was shown that error rates could be reduced significantly by training under multiple noise exposure conditions.

1. INTRODUCTION AND SUMMARY

Potential military applications of speech recognition systems often involve harsh environmental conditions and demanding tasks, where humans may be exposed to high ambient acoustic noise, encumbered by equipment such as an oxygen mask, and subjected to significant physiological and psychological stress [1,2]. Although current speech recognition technology can support restricted applications in benign environments and under low-stress conditions, recognition technology is not sufficiently advanced to provide robust, reliable performance in hostile and high-stress environments [1-6]. Difficulties include variabilities in the speech signal caused by stress and by exposure to noise in the speaker's ears [7,8], and additive noise at the input to the recognition system [9,10]. A number of efforts have recently been undertaken to quantify these problems [1-7, 9-12]. An important observation [12], which is in consonance with the results reported here, is that the effects on speech production of noise exposure at the ear (known as the Lombard effect [8]) appear to be more deleterious to recognizer performance than is the level of acoustic noise which passes through a noise-cancelling microphone. It also appears that other types of stress-induced variabilities (see [12], and results in this paper) have a more negative effect on recognizer performance than does additive noise.

This paper describes work carried out in the Speech Systems Technology Group of MIT Lincoln Laboratory, which is directed at the development of algorithms for robust, high-performance speech recognition in the fighter cockpit and other severe military environments.

An essential part of this effort, described in Section 2, has been the collection and analysis of a data base of speech produced under stress and in noise, and initial evaluation of the effects of the resulting speech variability on recognizer performance.

In developing techniques for robust recognition, we have chosen to build upon the Hidden Markov Model (HMM) approach [13-16]. (For additional references see the bibliographies in [13-15].) Reasons for choosing this approach include: excellent previously-reported recognition performance results under a variety of system constraints and in a variety of applications; effective extendability from isolated-word recognition to continuous speech recognition (although the experiments described in this paper are specifically focussed on isolated-word recognition); and trainability from observed data

by automatic methods. After a number of experiments with various forms of HMM, and preliminary investigations of potential HMM improvements [17], a baseline hMM recognizer was developed and implemented, as described in Section 3. This baseline recognition system was shown to perform well for speech produced under normal conditions, but to degrade significantly when presented with speech produced with variations typical in stress, or under noise exposure. Section 4 describes a number of robustness enhancements to the baseline HMM system, and a set of recognition results on a "simulated-stress" data base provided by Texas Instruments [12], which demonstrate the substantial improvements achieved using these enhancements. Section 5 describes an additional technique for improving recognizer performance, by compensation for stress effects [18] in the basic recognition input parameters, which are mel-frequency cepstral coefficients [19] in our system.

In Section 6 we focus on a technique called multi-style training [20] (also discussed in Section 3), which has been found to produce dramatic improvements in recognition accuracy under a variety of stress and noise exposure conditions. Recognition experiments are reported on a stressed-speech data base collected at Lincoln, where the recognizer is trained under stress conditi.ns simulated by variation in speaking style, and tested both for normal speech and under conditions of Lombard effect and perceptual/motor workload stress [21]. The results of these experiments were that multi-style training produced substantial error rate reductions relative to normal training for all conditions tested. Section 7 reports additional work on data collected in simulated F-16 noise conditions at the Air Force Medical Research Laboratory (AMRL). The data collected at AMRL includes simultaneous recordings from additional microphones mounted on the outside of the facemask, to be used in adaptive noise cancellation [22]. Experiments reported here were conducted on the primary microphone signal only. The results show that training under multiple conditions can substantially improve recognition performance, and indicate that the Lombard effect appears to be much more serious than additive noise in the recognizer input.

The statistically-based Hidden Markov Model is quite effective in its capability to absorb (by automatic training), and to make use of, the characteriscics of speech, in the context of an analytically-tractable model. Effective use of additional sources of acoustic-phonetic speech knowledge [23] should produce additional enhancements in recognition performance and robustness, particularly if we can build on the success already achieved using HMM approaches. Section 8 describes initial efforts in using feature-based discriminant analysis [24], to focus attention on the region of acoustic-phonetic distinction between pairs (or larger sets) of words which are observed to be difficult to distinguish with strict reliance on an HMM approach. As a further aid to improving recognition robustness in the fighter cockpit and similar environments, efforts (summarized in Section 9) have begun in utilizing articulatory

sensors including accelerometers and air pressure gauges to derive new acoustic parameters for input to the recognizer.

Finally, Section 10 summarizes conclusions from the work carried out up to now, and outlines areas for further work.

2. COLLECTION AND ANALYSIS OF A DATA BASE OF SPEECH PRODUCED UNDER STRESS AND IN NOISE

Psychological and physiological stress on a speaker lead to significant variations in the acoustic signal. Typical changes include: increased fundamental frequency, increase in the frequency and amplitude of the first formant, changes in overall spectral tilt, speech level and timing variations, and phonological modifications. Unfortunately, these vary significantly from speaker to speaker, and can be very different for the same speaker at different times. The effects of noise exposure (Lombard condition) have been observed [7] to produce similar changes to those produced under stress.

Collection of a large, systematic data base of speech produced under real stress conditions is a very difficult task. Our approach has been to obtain samples of speech produced under stress from a representative set of available sources, and to supplement this with a new data base developed in our Laboratory. This new data base, which we refer to as the Lincoln stress/style data base, includes speech produced during a difficult motor-workload task, under the Lombard condition, and with eight different talking styles designed to exhibit the range of acoustic variation typical of stressed speech.

The data base of speech produced under stress and noise which we have collected from other sources includes:

(1) the Advanced Fighter Technology Integrator (AFTI) F-16 data base [2,4,5], including effects of both noise and acceleration;

(2) sentences produced at AMRL by hazardous duty panel members before and after a drop tower run [25];

(3) two video tapes (with audio) made on F-16 aircraft during simulated combat exercises at Nellis Air Force Base [26];

(4) a tape containing the communication between a pilot and a controller just before a fatal helicopter crash [27];

(5) tapes made by 4 talkers during a task that induced vertigo [28].

Generally, the data showed the kind of variability expected for stress conditions. For some cases (e.g., drop tower) the changes seemed to be less than expected, while in other cases (helicopter crash) the changes were extreme. As an

82

example, Figure 1 shows spectrograms and parameters of two samples of a word recorded in an F-16 cockpit during simulated combat. The high stress sample shows increases in fundamental frequency, high frequency energy, frequency of first formant, and duration. Listening to the two recordings even indicates a dialect change, as the pilot's native drawl becomes more pronounced under the high-stress condition.

As we were beginning to develop our stress/style data base, we learned that researchers from Texas Instruments had independently produced an extensive "simulated stress" data base [12], including five talker styles (normal, fast, loud, soft, and shout) and the Lombard condition. The TI data base includes a 105-word "pilot" vocabulary, spoken by 5 male and 3 female talkers, with 5 training tokens (normal speech) per word per talker, and 2 test tokens per word per condition per talker. Their willingness to share this data base with us has greatly facilitated our research. Our recognition experiments and results on the TI simulated stress data base are described in Sections 4 and 5.

NORMAL AND HIGH-G/WORKLOAD "BROWN"
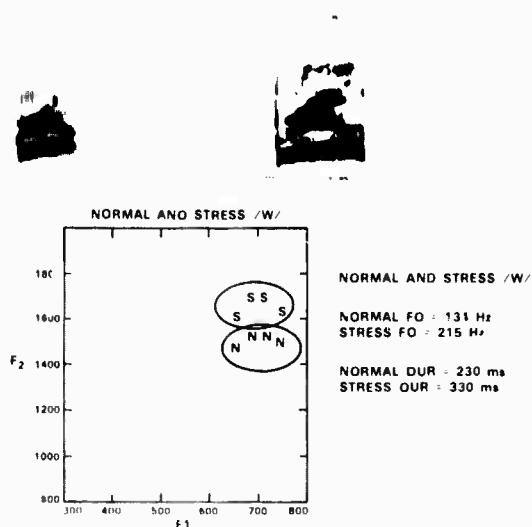
Fig. 1. Spectrograms and parameters of the word "Brown," recorded in flight in an F-16 cockpit under normal conditions and under high-G (≈ 2.5) and simulated combat workload conditions.

The Lincoln stress/style data base [29] contains 10,740 utterances produced by 9 talkers for 11 conditions: while performing a motor-workload task [21] (at two calibrated levels of difficulty) which has been used widely for workload research; under a Lombard condition (speech shaped noise presented binaurally at an overall level of 85 dB SPL); and with eight different talking styles (normal, slow, fast, soft, loud, clear enunciation, angry, and question pitch). The 35-word vocabulary

(which is a subset of the 105-word TI vocabulary) was selected to include a number of subsets which are difficult for recognition systems such as: {go, hello, oh, no} {six, fix} {while, wide}. Initial tests on this difficult vocabulary with a commercial recognizer [30] known to perform well on standard small-vocabulary data bases, indicated that error rates, which were relatively high (as expected) under normal conditions, increase sharply under workload stress, the Lombard condition, and under many of the style conditions. These tests provided some confirmation that currently-available speech recognition technology is not sufficiently robust to deal with a large range of stress-induced and noise-induced speech variations. Our recognition experiments and results on the Lincoln stress/style data base are described in Section 6.

3. BASELINE HMM SYSTEM DEVELOPMENT

The statistically-based Hidden Markov Model approach has, until recently, been applied most often to recognition tasks involving large vocabularies and/or continuous speech. The sustained effort and impressive results achieved at IBM [13,14] over the past 15 years exemplify this approach. Generally, limited vocabulary isolated-word-recognition (IWR) efforts had used a template-matching approach, combined with dynamic time warping (DTW) [31]. Initial applications of HMM to the IWR problem yielded results inferior to DTW approaches [32]; but later work [33] showed equivalent IWR performance for the two techniques when continuous parameters, rather than discrete, vector-quantized symbols, were used as the input to HMM. Recently, the HMM approach [16] has also been applied quite successfully in a commercially-available recognizer [30]. In addition to its good performance, the HMM approach has a number of advantages over DTW. These include: (1) better extendability to continuous speech than corresponding DTW techniques [34]; and (2) a convenient capability for automatic training on large amounts of data (including, for example, a number of tokens of each word). For these reasons, we have chosen HMM as our framework for developing robust recognition techniques.

HMM represents a family of techniques, rather than a single system, and therefore our initial efforts involved exploration and comparison of a number of HMM alternatives including both continuous-observation and discrete observation systems. After preliminary investigation of a new HMM training technique [35], a number of basic improvements to HMM techniques were developed and tested in preliminary form [17]. Discrete observation HMM systems generally use vector quantization [36] of the input speech parameters, with the vector quantizer being trained using the K-means [37] technique. A modified K-means technique was developed [17] for improved training of the vector quantizer in discrete observation HMM. Additional developments included: (1) improved smoothing of the observation probabilities in discrete observation HMM; and (2) tests of full durational models [38] and simplified durational models for the residency time in each state. Best results

[17] were obtained using a combination of these techniques.

That system, however, was deemed too complex for a baseline system. Therefore we applied some of our earlier work on continuous observation systems to define a baseline isolated-word HMM system, which is intentionally very simple and straightforward. The baseline system is a continuous-observation HMM, using mel-frequency cepstra [19] as its fundamental observation parameters. Specifically, the observations are the first 12 mel-frequency cepstral coefficients (without the energy term), computed every 10 ms. The joint probability density function of the cepstral parameters is assumed to be a multi-variate Gaussian distribution with diagonal convariance matrix (i.e., the individual cepstral coefficients for any frame of speech are assumed to be statistically independent). The word model network for each word is a linear sequence of independent nodes with no skip paths. A fixed number of nodes (ten) is used for each word. Since this system is intended to be used on speech files which consist of one word with some background (silence) at each end, the first and last nodes are background nodes to provide a semi-open-endpoint system. The system is trained using the forward-backward algorithm. The recognizer averages all initial background nodes and all final background nodes to prevent biases due to unequal endpoint nodes being carried over from training on files with varying smounts of background. A Viterbi decoder is used for recognition, and the highest probability word is chosen as the recognized word. Since the system assumes one word per file, only substitution errors are allowed.

Results obtained using the baseline HMM system and a number of enhanced systems, on speech produced under stress and in noise, are reported in the sections to follow.

All the experiments described in the sections to follow were performed using a Digital Equipment Systems VAX-11/780 with an attached Floating Point Systems array processor. This combination of processors requires about 20 seconds for each second of input speech to perform recognition on the 105-word vocabulary for the simpler systems (such as the baseline), and about 70 seconds for recognition using the full duration model. Computation time for training is also substantial. It should be emphasized that recognition processing requirements using Viterbi decoding with HMM will be similar to requirements for a DTW system with a similar distance metric. With current and emerging technology, it should not be difficult to develop a real-time HMM system of the type described here. However, processing power required to obtain good response times on the execution of recognition experiments on large data bases with a flexible facility is significantly larger than that required for real-time recognition.

4. ROBUST HMM SYSTEM DEVELOPMENT

A number of variations on the baseline system have been tried and some of them have yielded significant improvements in the recognition performance on the TI simulated stress ("TI-stress") database. The following presents the results of this work in three forms: a summary graph (Figure 2) showing results for some of the more important systems, a detailed table, and a textual description with comments. In the text, the results are presented in the form: (system1: x% vs. system2: y%). "System1: x%" represents the name and the average error rate of conditions 1-5 ("avg5" in the table) for the system currently under discussion; and "system2: y%" is the name and avg5 for the minimal pair system, to which we are comparing system1.
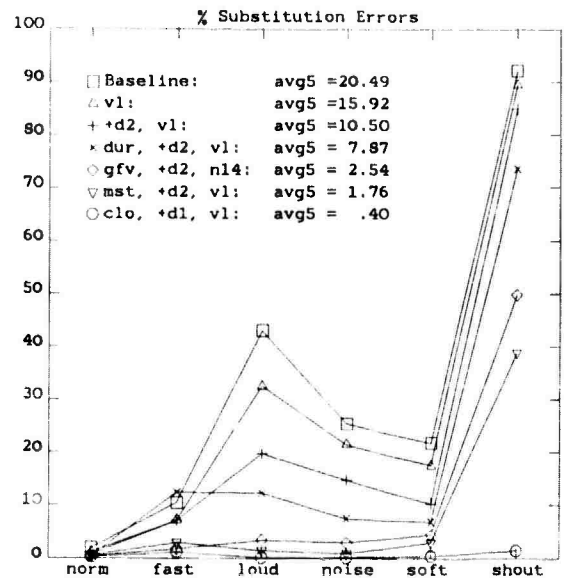

Fig. 2. Substitution error rates for the TI-stress data base. The codes for the various HMM recognition systems are defined in Table 1.

Avg5 is used for comparison rather than avg6 (the average error rate of all the test conditions), since the first five conditions are expected to be more indicative of speech from trained pilots than the shout condition. A full breakdown of the results, including both averages, is presented in Table 1.

The TI-stress data base [12] has a 105 word vocabulary, is recorded at a 4 kHz bandwidth, and consists of 8 speakers (5M + 3F) uttering 5 tokens of each word for enrollment, and 2 tokens of each word spoken in each of six conditions for testing. This gives a total of 1680 test tokens per condition. The conditions are: normal, fast, loud, Lombard (noise presented to the speaker in headphones, but not in the recorded speech), soft, an shout.

The baseline system uses 10 nodes (actually 10 active nodes plus a degenerate absorber) in a linear network (i.e., a path may only stay in the current node or move to the next node with no skips) and a trained diagonal covariance matrix for the multivariate Gaussian observation probability. (All systems described here use a diagonal covariance matrix.) The observations are mel-frequency cepstra [19]. This system yielded an error rate of 20.49%.

Increasing the number of nodes to 14 gave only an insignificant improvement in recognition performance (n14: 20.43% vs. baseline: 20.49%). Placing a lower bound on the variances (variance limiting, vl) yields a large improvement, probably because it corrects occasional gross underestimation of the variance as a result of the small training set (vl: 15.92% vs. baseline: 20.49%). The full duration model [38,15] (dur) also improves recognition results because it contains a more realistic duration model than does the standard HMM system (dur, vl: 10.10% vs. vl: 15.92%).

Significant improvements resulted from adding temporal difference parameters (+d1 for 10 ms and +d2 for 20 ms differences) to the standard observation parameter set (+d1, vl: 11.76% vs. vl: 15.92%) and for the duration model (dur, +d2, vl: 7.87% vs. dur, vl: 10.10%). The standard parameters contain only position and can convey motion only by moving to the next node. These difference parameters add the concept of motion to each individual node. Note that we use difference parameters in addition to the basic cepstral parameters, so that the number of observations per 10 ms frame increases from 12 to 24 for the +d1 and +d2 experiments. This additional information is used effectively by the system.

Dramatic improvements occurred with multi-style training (mst). The test database was split, and the first token of each word per condition was added to the training data, giving 11 tokens per word. The shout condition, even though it is not included in the avg5 error rate, was also used in the training. The second token was used for testing, giving 840 test tokens/condition. The standard observation (mst, vl: 3.48% vs. vl: 15.92%), differential observation (mst, +d1, vl: 2.21% vs. +d1, vl: 11.76%), and 20 ms differential observation (mst, +d2, vl: 1.76% vs. +d2, vl: 10.50%) systems all showed large improvements. Even the norm test conditions improved: .60% vs. 1.07%, .60% vs. .89%, and .60% vs. .65%, respectively, in spite of the added non-normal training data. Thus, the multi-style training not only helped the added styles, but also helped the norm condition which is similar to the standard training style.

Two multi-speaker systems have been tested. By "multi-speaker" system in this context, we mean that the system was trained by using speech from all eight speakers to yield a single HMM model per word for all eight speakers, and tested using different speech tokens from each of the same eight speakers. A normally-trained multi-speaker (msp)

system performed quite well compared to the corresponding speaker-specific system (msp, +d1, vl: 9.57% vs. +d1, vl: 11.76%), probably due to the large number of training tokens (40). It showed degradation in the better conditions (norm and fast) and improvement in the poorer conditions (loud, Lomb., soft, and shout). A second multi-speaker system using multi-style training was disappointing: (msp, mst, +d1, vl: 7.81% vs. mst, +d1, vl: 2.21%).

Some closed tests were performed using all test data for both training and test to estimate a lower bound on performance. As expected, the performance (clo, +d1, vl: .40%) was better than any of the open-test systems. While closed-test results are not indicative of operational recognition system performance, they do suggest that there is room for improvement. They also show the impressive ability of these HMM systems to incorporate widely varying speech styles into a single model.

The above systems have no explicit method of modeling phonological changes. A left-to-right (l-r) multi-style trained system which included single-node skip paths was tested to see if the additional freedom improved the recognition. The result was a decrease in performance (l-r, mst, vl: 4.64% error vs. mst, vl: 3.48%) Such a generalization requires more training data for accurate estimation of its parameters and it appears that we do not have sufficient data.

Several fixed diagonal covariance systems have also been tested. Note that the weighting given to each cepstral parameter in the distance computation are determined by the corresponding variance term on the diagonal of the covariance matrix (a smaller variance results in a larger weighting of the corresponding cepstral parameter). A unity variance matrix (L2 norm) system offered an improvement over the corresponding trained variance system (L2: 13.58% vs. vl: 15.92%). Somewhat better performance occurred when covariance derived from the enrollment data of all speakers was used (fvs: 8.76%). An even greater improvement was found when a covariance derived from perceptual considerations was used: (gfv: 6.13%). The improvement continued when differential parameters were added: (gfv, +d2: 4.99%). Finally, increasing the number of nodes to 14 produced the best error rate for a system trained only on the enrollment data: (n14, gfv, +d2: 2.54%).

Variance limiting, normal plus temporal differential observations, and multi-style training have been combined to improve the performance of the HMM recognition system. The variance limiting help to minimize the effects of limited training data and the differential observations give the models more useful information about the speech. The multi-style training, by including word-identification irrelevant variation, allows the models to focus on the invariant aspects of each word. The average error rate for the baseline HMM recognizer is 20%. The combination of the three techniques have reduced the average error rate by

an order of magnitude to 1.8%. Similar reductions
in error rate were also achieved without multi-
style training. A normally-trained system using a
perceptually-motivated, fixed-covariance distance
and normal plus differential observations achieved
an error rate reduction from 20% (baseline) to
2.5%.

	norm	fast	loud	Lomb.	soft	shout	avg5	avg6
Normal training (1680 tokens/condition):								
trained covar:								
baseline	1.90	10.48	42.98	25.36	21.73	92.20	20.49	32.44
n14	1.49	10.30	44.35	26.61	19.04	91.43	20.43	32.26
vl	1.07	7.14	32.50	21.43	17.44	89.52	15.92	28.18
+d1, vl	.89	6.79	22.38	16.07	12.68	86.31	11.76	24.19
+d2, vl	.65	7.14	19.76	14.70	10.24	84.94	10.50	22.91
dur, vl	.54	10.18	17.62	12.32	9.82	79.58	10.10	21.68
dur, +d2, vl	.36	12.44	12.26	7.44	6.85	73.57	7.87	18.82
fixed covar:								
L2	2.08	7.80	21.43	16.61	20.00	82.56	13.58	25.08
fvs	.95	5.48	17.44	10.71	9.23	76.79	8.76	20.10
gfv	.65	3.27	10.77	7.62	8.33	68.63	6.13	16.55
gfv, +d2	.36	2.32	7.68	4.82	9.76	59.17	4.99	14.02
gfv, +d2, n14	.36	1.73	3.39	2.86	4.35	49.76	2.54	10.41
multi-speaker, trained covar (msp):								
msp, +d1, vl	1.90	8.75	13.45	12.38	11.37	73.57	9.57	20.24
Multi-style training (mst) (840 tokens/condition):								
trained covar:								
mst, vl	.60	4.17	3.93	2.50	6.19	40.83	3.48	9.70
mst, +d1, vl	.60	2.86	2.38	1.79	3.45	35.95	2.21	7.84
mst, +d2, vl	.60	2.98	1.43	.83	2.98	38.81	1.76	7.94
multi-speaker, trained covar (msp):								
msp,mst,+d1,vl	2.98	8.45	7.26	8.21	12.14	46.19	7.81	14.21
left-to-right, trained covar:								
1-r, mst, vl	.83	4.52	6.07	3.21	8.57	41.19	4.64	10.73
Closed test, trained covar (1680 tokens/condition):								
clo, +d1, vl	.24	1.19	.12	.12	.36	1.43	.40	.58

TABLE 1

% SUBSTITUTION ERRORS FOR THE TI-STRESS DATA BASE:

vl=variance limiting
n14=14 nodes
+d1=added temporal difference parameters (10. ms)
+d2=added temporal difference parameters (20. ms)
dur=full durational model
L2=unity covariance matrix (equivalent to $L_2 >$ norm)
fvs,gfv=fixed variance
msp=multi-speaker
mst=multi-style trained
1-r=left-to-right model
avg5=average of conditions 1-6
avg6=average of all conditions

5. CEPSTRAL DOMAIN STRESS COMPENSATION

The success of the multi-style training experiments described in the previous section led us to investigate the comparative statistics of the cepstral parameters among the different conditions. Motivations were both to gain insight into the effects of different talking styles, and to investigate whether it would be possible to compensate for the cepstral changes through simple transformations on the cepstral means and variances obtained using normal training. Such transformations, if effective, could avoid the need for multi-style training.

The differences among normally-trained, single-style-trained, and multi-style-trained word models are partially reflected in the overall average shifts in the means and changes in the variances of the cepstral coefficients. To study such differences, we examined seven different sets of word models, trained under six individual conditions (normal, fast, loud, Lombard, soft, and shout), and under a composite of all these conditions (multi-style). The cepstral means and variances, averaged over all 105 words in the TI vocabulary and over all 10 nodes in each word, were computed for each of the models.

Figure 3(a) plots mean cepstral shifts (i.e., mean of the given model minus the mean of the normal model) for each of the cepstral coefficients. Shifts are shown for four cases: soft; shout; average of fast, loud, and Lombard; and multi-style. Figure 3(b) plots the corresponding spectra of these mean shifts, contrasting the effects on spectral tilt of low vocal effort (soft) versus higher vocal effort (fast, loud, Lombard, and shout). Increased vocal effort increases the relative high frequency content, whereas the opposite occurs with low vocal effort. It appears that these effects could be compensated, to some extent, by adding the appropriate cepstral compensation (Figure 3(a)) to normally-trained data.

Figure 3(c) plots the ratios of the cepstral variances of the multi-style-trained model to the cepstral variances of the normally-trained model. It appears that the major style-induced variations occur in the most slowly-varying spectral components (corresponding to lowest order cepstral coefficients) and in the most rapidly-varying spectral components (corresponding to the highest order coefficients).

A number of recognition experiments have been run to investigate the feasibility of improved recognition under varying conditions by means of cepstral domain compensation. In all these experiments, normally-trained word models were used, with mean and variance compensation (the same compensation for all words and nodes) applied according to smoothed versions of the data described above. Types of experiments included: (1) single-model compensation, where a set of cepstral mean differences observed in multi-style models (represented by filled squares in

Figure 3(a)) were applied as compensation in recognition tests on all styles; and (2) multi-model compensation, where four word models - corresponding to normal speech and to models compensated for low vocal effort, high vocal effort, and shout, were used for each vocabulary word. Note that the multi-model system requires a corresponding amount of extra computation to carry out the recognition. Error rates, averaged over the five conditions excluding shout, improved from 13.0% (for a baseline HMM system with varlim and additional spectral features [18]), to 9.7% for single-model compensation, and to 4.5% for multi-model compensation.

These results appear quite promising, although some issues remain, which are the subject of current investigation. First, cepstral domain compensation has not yet been combined with the other HMM improvements described in Section 4. Secondly, an experiment should be performed where the cepstral statistics are gathered on a data base separate from that used in recognition tests. In addition to investigation of these issues, we are currently pursuing some new and promising ideas in cepstral domain stress compensation, which do not require multiple word models, or computation of cepstral statistics over a large data base.

(a) mean shifts in cepstral domain

(b) spectra of cepstral mean shifts (dB scale)

+ soft
Δ multi-style
□ average of fast, loud, and Lombard
o shout

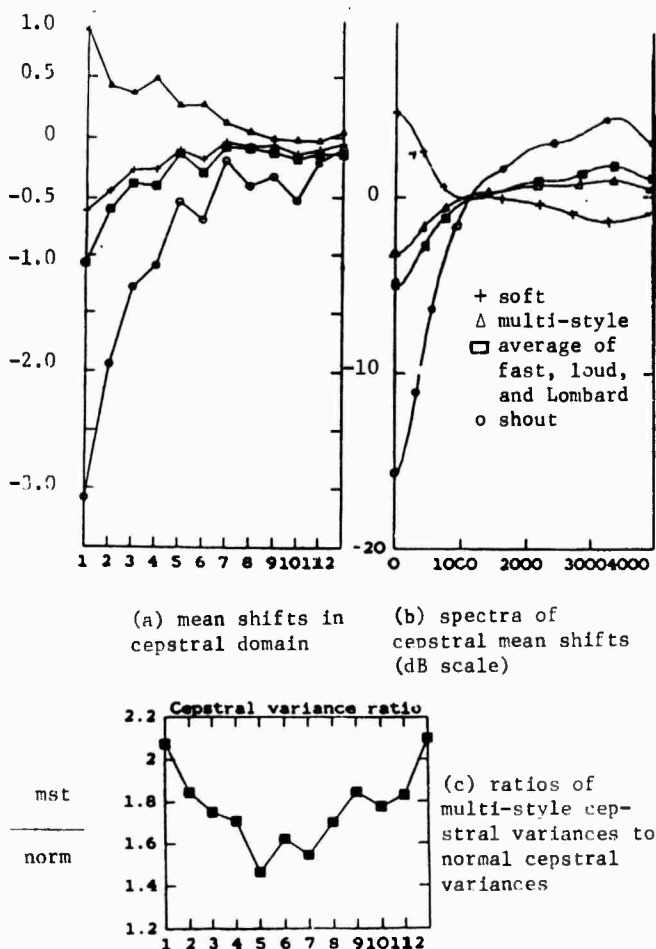(c) ratios of multi-style cepstral variances to normal cepstral variances

Fig. 3. Variations of cepstral means and variances for simulated stress and Lombard conditions.

6. MULTI-STYLE TRAINING FOR IMPROVED RECOGNITION UNDER WORKLOAD STRESS AND UNDER THE LOMBARD CONDITION

Multi-style training can be applied effectively to the recognition-in-stress problem by training the system under a variety of talking styles and/or a variety of conditions, and then using the system under stress conditions. Training tokens can be obtained under the same conditions experienced during use (if possible), or they can be obtained by instructing talkers to speak with different styles. For the results presented in Section 4 on the TI-stress data base, training tokens were obtained under the same set of conditions (style and Lombard) as those used for testing. This strategy is not possible in many operational conditions, such as in a fighter cockpit, where training would have to be performed during landing and during times of high workload. In this section, results are presented for the more general situation where training tokens are not available under testing conditions.

All experiments were performed using the Lincoln stress/style data base, and the baseline HMM recognition system with variance limiting. The recognizer was trained with normal speech (five training tokens for each vocabulary word), and using multi-style training (one normally spoken training token, and one token each from the fast, clear, loud, and question pitch style conditions). The recognizer was then tested using normally-spoken words (different tokens from those used in training), and using speech produced under the Lombard condition and under the two workload stress conditions. Each condition was tested using 70 tokens (two samples of each vocabulary word). Results for five talkers from the Lincoln data base are presented in Figure 4.
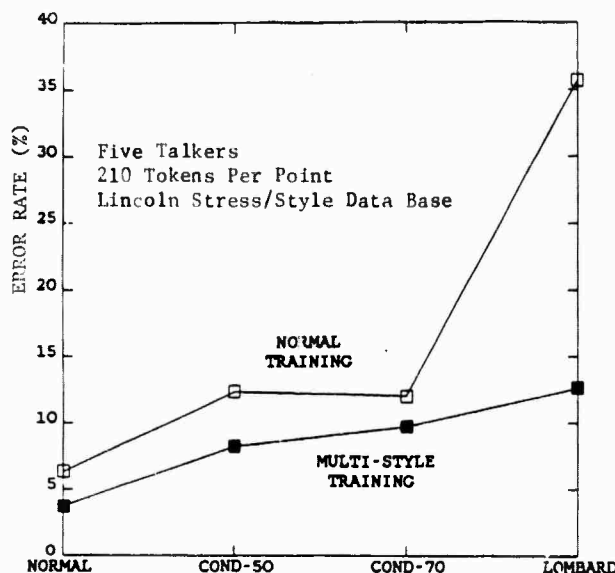

Fig. 4. HMM (vl) system performance on Lincoln workload stress and Lombard condition speech data, with normal training and with multi-style training.

The results demonstrate that multi-style training can provide a large performance advantage under stress conditions, without training under those conditions. Multi-style training reduces the error rate by more than a factor of two under the Lombard condition (talker in noise), and provides large reductions in error rates under other conditions. An interesting, and perhaps surprising result is that error rates also dropped substantially for normal speech. The fact that overall error rates in Figure 4 are generally significantly higher than in Figure 3 can be attributed to the difficult conditions the highly confusable vocabulary.

The large reductions in error rate with multi-style training are presumably due to the fact that the forward-backward training algorithm is able to collect better statistics concerning the variability of different acoustic-phonetic features. Performance improves for normal speech presumably because there is insufficient variability in the five normally-spoken training words to characterize the variability that occurs in normal speech spoken over a period of a few days. Performance improves under workload stress and with the Lombard condition (1) because the multi-style training conditions provide a better match to the test conditions than does normally-spoken speech; and (2) because the HMM recognizer seems to focus on those characteristics of speech that are invariant across talking styles.

7. MULTI-CONDITION TRAINING FOR IMPROVED RECOGNITION IN SIMULATED F-16 NOISE ENVIRONMENT

Background noise in a fighter cockpit has two major effects. First, it causes the pilot to speak louder and more distinctly (the so-called Lombard effect). Second, it leaks into the microphone, mixes in with the speech signal, and degrades the input signal-to-noise (S/N) ratio. The relative importance of these effects was investigated recently using recordings made at AMRL. Words in the 25-word AFTI vocabulary were produced by one talker wearing a facemask and helmet in an ambient condition and with simulated AFTI F-16 background noise levels of 95 dB, 105 dB, and 115 dB sound pressure level (SPL).

Recognition experiments were conducted using our baseline HMM isolated-word recognizer with variance limiting. Experiments were carried out with normal training (five tokens from the ambient condition) of the recognizer, and with a new type of training which we refer to as multi-condition training (two tokens from the ambient condition and one from each noise condition). Multi-condition training is distinguished here from multi-style training in that training is done by subjecting the talker to different noise exposure conditions, rather than asking the talker to speak with a variety of talking styles. Results are presented in Figure 5. Signal-to-noise ratios presented at the top of this figure are obtained by determining the ratio between the 95 percent and 5 percent

cumulative RMS levels measured across the waveform file for each word, using a 100 ms rectangular averaging window to determine RMS.

As can be seen in Figure 5, with normal training the error rate ranges from zero in the ambient condition to greater than 60 percent with 115 dB SPL of background noise. Multi-condition training, however, provides a dramatic improvement in performance. The error rate is reduced to zero except for the 115 dB SPL condition. Objective measurements of the S/N ratio (noted in Figure 5) and careful listening to the recordings strongly suggest that degraded performance was caused by the Lombard effect, and not by additive noise. These results, which are consistent with those presented in [12], thus indicate that research emphasis should be placed on compensating for the Lombard effect and not on compensating for a degraded S/N ratio. Some compensation for a degraded S/N ratio will still be necessary under very high noise conditions, but such compensation will not be sufficient for good recognition. These results also demonstrate that multi-condition training in a HMM recognizer is an effective technique to compensate for the Lombard effect.
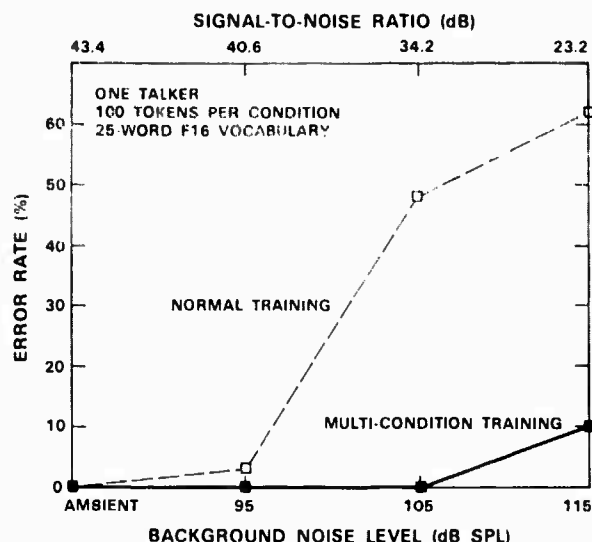
Fig. 5. HMM system performance in simulated F-16 noise environment, with normal training and with under multiple noise conditions.

8. FEATURE-BASED DISCRIMINANT ANALYSIS

Most isolated-word speech recognition systems weigh all parts of a word equally when comparing an input unknown word to stored-word models. This may lead to errors for words such as "go" and "no" that differ only in one temporal region. Discriminant analysis [24], as described here, is a technique that can overcome this problem by focussing attention on parts of words determined to be most important in discriminating between a small set of word candidates. The approach described here is similar in some respects to that proosed in [39].

A different approach to discriminant analysis, where analysis of recognition errors is used to select an optimum set of "features" used to characterize spectral characteristics, is described in [40].

We propose to build on our successful HMM systems and use discriminant analysis in a second analysis stage. The first stage of analysis will include a maximum-a-posteriori probability Hidden Markov Model recognizer. The output of this stage will consist of overall word probabilities, node residency times from the Viterbi backtrace, and information concerning the distribution of spectra in each node. The second discriminant stage will use this information, along with information from additional acoustic-phonetic feature measurements designed to improve discrimination decisions between specific word pairs. Statistics required for the discrimination stage will be obtained during training by: (1) passing each training word through feature measurements to determine statistics of these measurements; and (2) passing each training token through the word models for all vocabulary words, rather than only through the word model for the corresponding word.

Our plan is to carry out a second-stage discriminant decision only if likelihoods from the first-stage HMM recognizer indicate that no clear decision can be made. This strategy is designed to prevent the second stage analysis from degrading the good performance of existing HMM recognizers. Results obtained with the Lincoln stress/style data base with multi-style training for the Lombard and workload conditions suggest that a relatively simple decision rule based on the difference between the log likelihoods of the first two word candidates is effective in determining whether to perform a discriminant analysis. This rule correctly identified roughly 90 percent of the trials where an error occurred and incorrectly identified 24 percent of those trials where the HMM recognizer made a correct decision.

A second-stage discriminant decision will initially be performed using a two-way discriminant between only the top two word models. Data shown in Figure 6 illustrate that a two-way discriminant could substantially reduce error rates. Data in this figure represents all the multi-style training conditions in Figure 4. Figure 6 indicates that perfect discrimination between the top two word candidates would reduce the error rate by more than a factor of two. N-way discriminations between the top three or more candidates would provide smaller incremental reductions in error rate.

Initial results with the above discriminant strategy, using only duration information as a discriminant, have been encouraging. Further work is currently in progress that uses spectral features, computed from the cepstral parameters already available in the baseline system, in the discriminant decision.
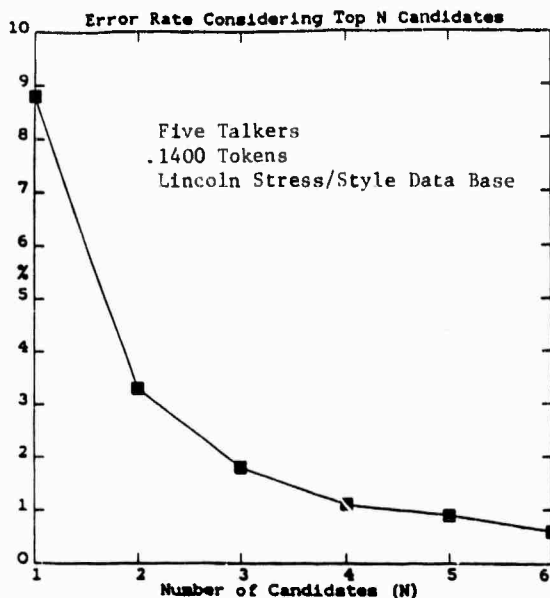
Fig. 6. HMM system error rate considering the top N word candidates, using multistyle training on the Lincoln stress/style data base.

9. APPLICATION OF ARTICULATORY SENSORS

The goal of experiments with articulatory sensors is to determine whether the outputs of small, non-invasive sensors can be used to supplement the microphone signal and improve speech recognition performance in noise and stress. Rather than focussing on obtaining a signal that is more noise-free than a microphone signal, our primary focus here is on obtaining a signal that provides additional information concerning articulatory movements which are consistent for specific phonetic events. Two types of sensors will be used: miniature accelerometers and a static pressure gauge transducer. Miniature accelerometers can be positioned on the nose [41], throat [42], and forehead of a talker to detect voicing energy and nasal energy. In a cockpit, the nasal accelerometer could be mounted in the soft rubber of a facemask and the accelerometer on the forehead could be mounted in the helmet on a foam pad. The static pressure gauge monitors pressure within the facemask via a piece of short flexible tubing that runs through the facemask. It can be used to detect the rapid increase in pressure associated with plosives [43].

A sketch of a multi-sensor system, including articulatory sensors and a secondary microphone for possible application in adaptive noise cancellation, is shown in Figure 7. Preliminary articulatory sensor recordings have been made by simultaneously sampling the outputs of three accelerometers (nasal, throat, forehead) of the static pressure gauge, and of the facemask microphone. After creating a small multi-sensor data base, we will investigate the feasibility of processing the outputs of these sensors to provide features which can enhance recognition robustness.
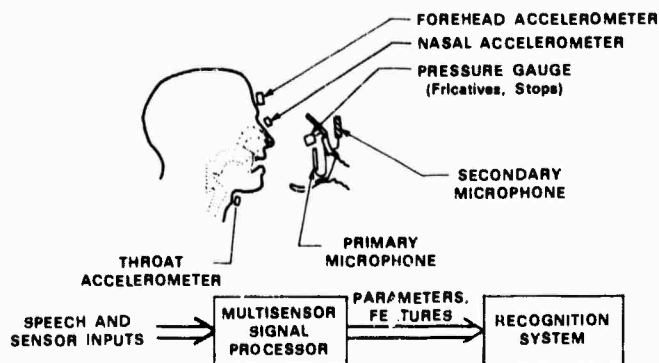

Fig. 7. System structure for potential application of multisensor signal processing to robust speech recognition.

10. CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

Substantial improvements in speech recognition performance on speech produced under stress and in noise have been demonstrated, through the development of techniques for enhancing the robustness of a baseline isolated-word HMM recognizer. The performance achieved - about 2% error rate for a 105-word vocabulary, under a variety of stress and noise conditions - should be sufficient to support a reasonable range of speech recognition applications in stressful environments, where limited vocabularies and speaker-specific training can be used. This performance represents an order-of-magnitude reduction in error rate relative to a baseline HMM system, and compares favorably with the best results previously reported using DTW techniques. The capability of the HMM system to train effectively on multiple talking styles, and thus to become more tolerant to speech variations due to stress and noise exposure, is an extremely encouraging result.

Based on the results obtained so far, we see a large number of areas for current and projected future work in robust recognition, including:

(1) more work on HMM robustness improvements, including augmented parameter sets, durational models, better use of energy, minimization of the effects of limited training data, and use of word models with different number of nodes for different vocabulary words;

(2) further development of cepstral domain stress compensation techniques;

(3) extensive tests of robust recognition techniques on additional data bases, including physical stress, and eventually data collected in flight;

(4) development and test of techniques, including discriminant analysis, for integration of acoustic-phonetic features with HMM systems;

90

(5) investigation of feature extraction from articulatory sensor data;

(6) dynamic adaptation techniques for updating training while the recognition system is in use;

(7) extensions from isolated-word to connected-word recognition systems in the context of high stress and noise, including subword HMM models;

(8) efforts in talker-independent recognition, taking advantage of the techniques for dealing with speech variability that have been developed for the recognition-in-stress problem; and

(9) eventual integration into application systems, and test on-board high-performance aircraft and in other severe environments.

Finally, although our efforts have focussed on the stress and noise problem, we feel that the techniques applied to deal with the variations in speech due to stress should be applicable to more general recognition problems.

REFERENCES

[1] "Automatic Speech Recognition in Severe Environments," report prepared by the Committee on Computerized Speech Recognition Technologies for the Commission on Engineering and Technical Systems of the National Research Council, published by National Academy Press, Washington, D.C., 1984.

[2] T.R. Anderson and T.J. Moore, "Characterizing Requirements of Speech I/O in Military Cockpit Environments," Proceedings Speech Tech 84, April 1984.

[3] J.W. Armstrong and G.K. Poock, "Effect of Operator Mental Loading on Voice Recognition System Performance," Naval Postgraduate School Technical Report NP555-81-017, Monterey, CA, 1981.

[4] D.T. Williamson and D.G. Curry, "Speech Recognition Performance Evaluation in Simulated Cockpit Noise," Proceedings Speech Tech 84, Media Dimensions, Inc., April 1984.

[5] E. Werkowitz, Speech Recognition in the Tactical Environment: the AFTI/F-16 Voice Command," Proceedings Speech Tech 84, April 1984.

[6] C.R. Coler, "In-Flight Testing of Automatic Speech Recognition Systems," Proceedings Speech Tech 84, April 1984.

[7] D. Pisoni, R.H. Bernacki, H.C. Nusbaum, and M. Yuchtman, "Some Acoustic Phonetic Correlates of Speech Produced in Noise," Proceedings of ICASSP '85, pp. 1581-1584.

[8] E. Lombard, "Le Signe de l'Elevation de la Voix," Ann. Maladiers Oreille, Larynx, Nez, Pharynx, Vol. 37, 1911, pp. 101-119.

[9] G. Neben, R.J. McAulay, and C.J. Weinstein, "Experiments in Isolated Word Recognition Using Noisy Speech," Proceedings of ICASSP '83, April 1983.

[10] E.J. Cupples, "Speech Enhancement for Robust Recognition," 5th Western Conference and Exposition, AFCEA, January 1984.

[11] P.K. Rajasekaran and G.R. Doddington, "Speech Recognition in the F-16 Cockpit Using Principal Spectral Components," Proceedings of ICASSP '85, pp. 882-885.

[12] P.K. Rajasekaran, G.R. Doddington, and J.W. Picone, "Recognition of Speech Under Stress and in Noise," to be published in Proceedings of ICASSP '86.

[13] F. Jelinek, "Bibliography on the Use of Probabilistic Models in Speech Recognition," distributed at the IEEE-ASSPS Workshop on Frontiers of Speech Recognition," Arden House, Harriman, NY, December 1985.

[14] F. Jelinek, "The Development of an Experimental Discrete Dictation Recognizer," Proceedings IEEE, Vol. 73, No. 11, pp. 1616-1624, November 1985.

[15] S.E. Levinson, "Structural Methods in Automatic Speech Recognition," Proceedings IEEE, Vol. 73, No. 11, pp. 1625-1650, November 1985.

[16] J.K. Baker, "The Dragon System - An Overview," IEEE Trans. ASSP, Vol. ASSP-23, No. 1, pp. 24-29, February 1975.

[17] D.B. Paul, in preparation.

[18] Y. Chen, in preparation.

[19] S.B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," IEEE Trans. ASSP, Vol. ASSP-28, No. 4, pp. 357-366, August 1980.

[20] R.P. Lippmann, M.A. Mack, and D.B. Paul, "Multi-Style Training for Robust Speech Recognition in Stress," to be presented at the Acoustical Society of America Conference, May 1986.

[21] H.R. Jex, "A Proposed Set of Standardized Sub-Critical Tasks for Tracing Workload Calibration," in N. Moray, Mental Workload: Its Theory and Measurement, New York: Plenum Press, pp. 170-188 (1979).

[22] W.A. Harrison, J.S. Lim, and E. Singer, "A New Application of Adaptive Noise Cancellation," IEEE Trans. ASSP, Vol. ASSP-34, No. 1, February 1986.

[23] V.W. Zue, "The Use of Speech Knowledge in Automatic Speech Recognition," Proceedings IEEE, Vol. 73, No. 11, pp. 1602-1615, November 1985.

[24] R.P. Lippmann and E.A. Martin, in preparation.

[25] Provided by T. Moore and T. Anderson, AMRL.

[26] Provided by Major J. Severski, Nellis AFB.

[27] Provided by M. Brenner, San Francisco Veterans Administration.

[28] Provided by K. Stevens, MIT.

[29] R.P. Lippmann, internal note.

[30] J.K. Baker, J.M. Baker, R. Roth, and P. Bamberg, "Cost-Effective Speech Processing," ICASSP '84 Conference Record.

[31] F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans. ASSP, Vol. ASSP-23, pp. 67-72, February 1975.

[32] L.R. Rabiner, S.E. Levinson, and M.M. Sondhi, "On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition," Bell System Technical Journal, Vol. 62, No. 4, pp. 1075-1105, April 1983.

[33] L.R. Rabiner and B.H. Juang, "An Introduction to Hidden Markov Models," IEEE ASSP Magazine, Vol. 3, No. 1, pp. 4-16, January 1986.

[34] C.S. Myers and L.R. Rabiner, "A Level-Building Dynamic Time Warping Algorithm for Connected Word Recognition," IEEE Trans. ASSP, Vol. ASSP-29, pp. 284-297, 1981.

[35] D.B. Paul, "Training of HMM Recognizers by Simulated Annealing," Proceedings of ICASSP '85, pp. 13-16, April 1985.

[36] J. Makhoul, S. Roucos, and H. Gish, "Vector Quantization in Speech Coding," Proceedings IEEE, Vol. 73, No. 11, pp. 1551-1588.

[37] M.R. Anderberg, Cluster Analysis for Applications, New York, NY: Academic Press, 1973.

[38] J.D. Ferguson, "Variable Duration Models for Speech," in Proc. Symp. on Application of Hidden Markov Models to Text and Speech, J.D. Ferguson, Ed., Princeton, NJ, pp. 143-179, 1980.

[39] L.R. Rabiner and J.G. Wilpon, "A Two-Pass Pattern Recognition Approach to Isolated Word Recognition," Bell System Technical Journal, Vol. 60, No. 5, pp. 739-765, May 1981.

[40] E. Bocchieri and G.R. Doddington, "Frame-Specific Statistical Features for Speaker Independent Speech Recognition," to be published in IEEE Trans. ASSP.

[41] R.P. Lippmann, "Detecting Nasalization Using a Low-Cost Miniature Accelerometer," J. of Speech and Hearing Research, Vol. 24, pp. 314-317, September 1981.

[42] V.R. Viswanathan, et. al., "Multisensor Speech Input for Enhanced Immunity to Acoustic Background Noise," Proceedings ICASSP '84.

SECTION 3

REVIEW OF SUPPORTING INFRASTURCTURES

THE DARPA SPEECH RECOGNITION RESEARCH DATABASE:
SPECIFICATIONS AND STATUS

William M. Fisher
George R. Doddington
Kathleen M. Goudie-Marshall

Texas Instruments Inc.
Computer Sciences Center
P.O. Box 226015, MS 238
Dallas, Texas 75266, USA
Tel. (214) 995-0394

ABSTRACT

This paper describes general specifi-cations and current status of the speech databases that Texas Instruments (TI) is collecting to support the Darpa speech recognition research effort. Emphasis is placed on the portion of the database development work that TI is specially responsible for. We give specifications in general, our recording procedures, theoretical and practical aspects of sen-tence selection, selected characteristics of selected sentences, and our progress in recording.

1. INTRODUCTION

This paper is a report on the speci-fication and current status of the work done by Texas Instruments, Inc. (TI) on Darpa-funded Acoustic Phonetic Database development as of the early part of February, 1986. It is meant to be comple-mentary to similar reports from other groups included in this volume.

2. GENERAL SPECIFICATIONS

Originally three data bases were planned: "stress," "acoustic-phonetic," and "task-specific." The stress data base was to investigate variations of speech with stress, and would be done primarily by AFAMRL. The acoustic-phonetic data-base, to be done by TI in collaboration with MIT and SRI, was intended to uncover general acoustic-phonetic facts about all major dialects of continental U.S. English. And the task-specific data base, providing data for the study of the effect on speech recognition of limiting domain of discourse, would be defined later. At our last meeting, there was a consensus that the task-specific data base should be folded into the acoustic-phonetic data base, becoming one of the later phases.

The acoustic-phonetic data base is phased so that a small amount of speech is initially recorded from a large number of subjects, followed by successively larger durations of speech from fewer subjects, culminating in two hour recorded from each of two subjects. MIT and SRI have helped us design the material to be read by subjects. Figure 1 below shows the current general specifications for this data base.

3. RECORDING PROCEDURES

3.1 STEROIDS

This large scale database collection would be difficult or impossible to col-lect without the VAX Fortran automated speech data collection system developed here at TI, called the STEReO automatic Interactive Data collection System, or STEROIDS. Use of STEROIDS requires a stereo DSC 200 sound system directly connected to 2 DSC 240 audio control boxes, one for each of the 2 channels of stereo input. (No multiplexor is used.)

STEROIDS uses a file called the Vocabulary Master Library (VML). The VML file is a formatted direct access file which contains records holding data for each utterance in a recording session: the text of the prompt, a speech file name, and variables holding the number of recorded versions and which one is best. A prompt may be any text string less than 133 characters long.

When STEROIDS is executed, it first reads in values for several parameters that effect its decisions about when each utterance begins and ends, and a name for the subject. It then, under the control of the director, displays prompts to the subject and records his responses in speech files. The director may listen to recorded versions, decide which version is best, and re-prompt.

Recording conditions:
  o Low noise (acceptable to NP^)
  o 2 channel recording: 1 noise-cancelling (Sennheiser) mike,
    1 far-field pressure (Bruel and Kjaer) mike.
  o Subjects exposed to 75 dB SPL noise through earphones

Style:
  o Read from prompts

Material:

Phase	Speech/Subject	# Subjects	Contents, etc.
1	30 sec.	630	Broad Phonetic Coverage
2	2 min.	160	
3	8 min.	40	W/Standard Paragraph
4	30 min.	10	W/Explicit Variations
5	2 hrs.	2	Interview Format

Figure 1. General Specifications of Acoustic-Phonetic Database.

3.2 GENERAL PROCEDURE

We created and ran a program which read sentences and sentence assignments and made 630 VML files. Our recording procedure then takes five steps: 1. At the beginning of each day, calibration tones are recorded from both channels; 2. For each subject, one of the 630 VML files is copied to his named sub-directory and STEROIDS is used to collect his data; 3. At the end of the day, a REDUCE procedure is run on all data collected that day, which produces the files that we send out, by splitting the initial stereo file into two mono files, de-biasing each, high-pass filtering the BK file at 70 ..., and down-sampling each to 16,000 samples per second; 4. A backup procedure is then run, which makes three tape copies of the VML files, the calibration tone files, and all the speech files recorded on that day; and 5. The disk is cleaned up for re-use by deleting the files that were put onto tape. One copy of the back-up tape is then sent to NBS.

Data on each subject recorded in each session is added to an ASCII text file for documentation.

3.3 NOISE

After the sound booth was moved to the third floor of the North Building, a very large noise signal was observed coming from the combination BK power supply and preamplifier. At first this noise was thought to be the result of a defect in the amplifier, but the BK service center could find no problem. It was then that we realized that the noise was actually an acoustical signal being picked up by the microphone. Figure 2 shows the spectrum of the noise signal

below 500 Hz for a 5 second segment of "silence". The spectrum is flat from 300 Hz up to 10 kHz. (The spectrum of the signal from the Sennheiser noise-cancelling microphone is flat from DC to 10 kHz, which indicates that the noise-cancellation property and the low-frequency roll-off of the Sennheiser is adequate to render the acoustic rumble of no consequence for this microphone.)
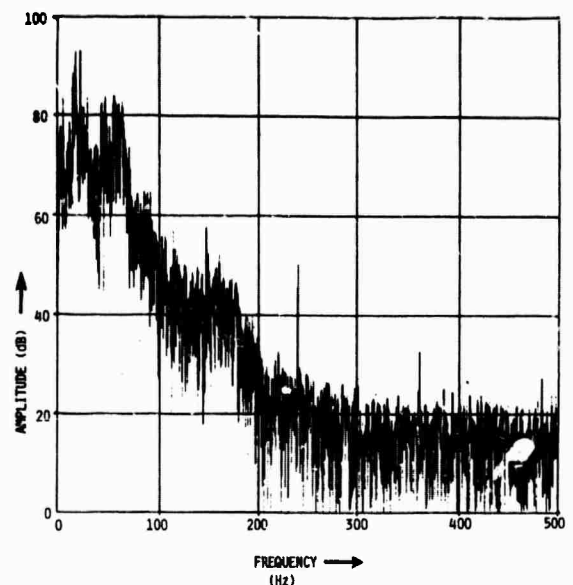

Figure 2. Amplitude Spectrum of Acoustic Rumble. Recorded in the TI sound booth over a 5 second period of "silence."

With consultation from an acoustical engineering consultant it was judged that the acoustical noise in our double-walled sound booth is being introduced by mechanical vibrations transmitted through the floor. Opinion varies as to the amount of reduction that may be achieved by better isolation from the floor, from less than 3 dB to more than 20 dB. Current plans are to install an air suspension vibration isolation mount system under the sound booth to reduce the rumble as much as possible.

As an interim solution, a 1581-point FIR filter has been designed to provide a high-pass filter function, with a cut-off at 70 Hz and an in-band ripple of less than 0.1 dB above 100 Hz. Using this filter, reasonably acceptable S/N ratios have been achieved during data collection. The following S/N ratios have been measured, using seventeen subjects' (nine men and eight women) utterances of sentence SA1.

Condition	ENrms	SNavg	SNpk
No HP	421	8 dB	16 dB
70 Hz HP	95	21 dB	29 dB
200 Hz HP	4	48 dB	56 dB

Table 1. Raw S/N Ratios.

Explanatory notes for Table 1: ENrms is the RMS energy of the noise; SNavg is the average S/N ratio, signal energy being computed as the average RMS signal value over the entire utterance; SNpk is the peak S/N ration, signal energy being computed as the peak RMS signal value in a 30 msec. Hamming-weighted window slid across the utterance.

During this tabulation it was noticed that the RMS energy for men's utterances averaged 4 dB greater than that for women. There are variations of signal level with speaker and with utterance, of course, and the weakest of the seventeen utterances used for this tabulation showed an average S/N ratio of minus 1 dB for the original signal.

The effective S/N ratios for speech processing and listening or perceptual purposes will be somewhat higher for the no high-pass and 70 Hz high-pass results listed above, because typically a pre-emphasis is performed on the speech signal before further processing, and because the human ear is progressively less sensitive to sound at frequencies below 200 Hz. For a preemphasis constant of 1.0 (at a sampling frequency of 16 kHz), the S/N ratios were measured as follows:

Condition	ENrms	SNavg	SNpk
No HP	6	35 dB	46 dB
70 Hz HP	4	39 dB	50 dB
200 Hz HP	3	41 dB	52 dB

Table 2. Pre-emphasized S/N Ratios. Symbols are same as in Table 1.

4. ACOUSTIC-PHONETIC DATA BASE PHASE 1

4.1 GENERAL

The sentences constituting the phase 1 material will have a mean value of expected reading time of 3 seconds, so that each of the 630 subjects reading ten sentences will give us the specified 30 seconds per subject of speech data.

Altogether 630x10=6300 sentence tokens will be collected. The sentence types are divided into three sorts: 1. Two "dialect" or "calibration" sentences; 2. 450 "MIT" sentences; and 3. 1890 "TI" sentences. Each subject reads both the dialect sentences, a selection of five of the MIT sentences, and a selection of three TI sentences. Each MIT sentence will be read by seven speakers and each TI sentence by one. This variation in the number of subjects reading different sentences is a compromise between the desiderata of breadth and depth of phonetic coverage across subjects.

The dialect sentences were devised by SRI and the MIT sentences by MIT, who will report separately on their design.

4.2 THE TI NATURAL PHONETIC SENTENCES

Our strategy in selecting our 1890 sentences was almost identical to one we reported on earlier [1]: use a computer procedure to select from a large or infinite set of sentences a subset that meets certain feasibility criteria, trying to optimize an objective function of the selected sentences. The ideal set of sentences to draw from in this case is the set of normal, acceptable American English sentences. Lacking an off-the-shelf grammar of sufficient generality, we approximate this set with the largest set of American English sentences in computer readable form that we know of, the "Brown Corpus [2]." Responding to concerns of some in the DARPA Database SIG that these sentences were "written" English instead of "spoken" English, we augmented our final pool of sentences from this corpus with 136 sentences of playwrights' dialog from the corpus published by Hultzen et al. [3]. (We are not concerned that our sentences are too "written": the alternative, naturally "spoken" sentences, are replete with run-on sentences, self-corrections, and ungrammaticality.)

There may be some slight discrepency between the original written form of these Hultzen sentences and the form in which we use them, since we reconstructed their spellings from the phonemic transcriptions published in the Hultzen book using TI off-the-shelf speech-to-text technology.

A series of programs was executed that produced a file of pointers to the beginnings of sentences in the Brown corpus, then filtered out sentences from this set until about 10,000 were left in the selection pool. Sentences were eliminated if they were over 80 characters long, included any proscribed words, or included characters other than letters and punctuation. This pool was augmented with 136 Hultzen sentences.

The fixed set of sentences -- the two dialect sentences and the revised set of 450 sentences that TI received from MIT in the middle of November -- were transcribed phonemically by TI's best off-the-shelf text-to-phoneme program and, after careful checking by two experts in phonetics and phonology, files of allophonic transcriptions of them were computed as described below. The selection program assumed this set of utterances as a base to build on in the selection of the 1890 TI sentences. The selection pool of 10,000 sentences was prepared in a similar way, except that it was not feasible to hand-check the transcriptions.

The selection program accesses these allophonic transcription files, in addition to a file of pointers to sentences that have previously been selected and one of pointers to sentences that have been manually zapped (ruled out). It produces a new version of the sentence selection file. Both the sentence selection file and the zapped sentence file are in ASCII text file format so that they can be manipulated with a text editor. One of the program's typed-in parameters tells it how many sentences to select. The program was run in a series of batch jobs, each typically selecting an additional 100 or so sentences. The additional sentences selected in each batch run were examined, and unacceptable ones were stricken from the selected sentence file and added to the zapped sentence file before the next run.

The internal procedure used by the program is this:

1. Build the initial version of the data structure holding phonetic data on the selected sentences by reading in the dialect sentences weighted by 530, the MIT sentences weighted by 7, and the previously selected TI sentences weighted by 1;

2. Repeat this until this run's quota of sentences has been selected:

a. Scan through a list of prospective sentences from the pool of unselected and unzapped sentences, calculating for each the increase in the phonetic objective function under the hypothesis that the sentence is added to the selected set, remembering the one producing the highest value;

b. Add the remembered sentence to the selected sentence list.

3. Write out the new version of sentence selections.

The program knows two basic ways of making a list of sentences from the pool for examination: 1. take N (typically 400) random grabs; and 2. look at them all. This option is selectable by the user, and both were used in actual runs selecting sentences. The first is faster and less optimal than the second.

4.3 CONTROL OF AVERAGE UTTERANCE DURATION

In order to control the average duration of utterances, a heuristic was used. The expected speech duration of each sentence was calculated using the formula

$$SPDUR = -0.0928 + .06302 * NLETTS$$

where NLETTS is the number of letters in the spelling of the sentence and SPDUR is the speech duration of the sentence in units of seconds. This formula was derived by the least-squared-error fit of a linear function to speech duration data obtained from a previously collected data base of continuous speech: 750 sentences from each of eight subjects, half male and half female. The mean value of speech utterance duration of the current selected sentence set was kept track of, and if it was lower than the target duration (three seconds) minus a tolerance, the next sentence selection was taken from a list of longer-than-average pool sentences; if the mean speech duration was greater than the target plus a tolerance, the next selection was from the subset of short pool sentences; and if within the tolerances, any of the 10,000 pool sentences could be selected. The tolerance used in the final selection was 1%.

4.4 OBJECTIVE FUNCTION

The function that is used to measure the aggregate phonetic coverage of the set of selected utterances, called "allophone information", is:

$$Ial = SUM(Ni*LOG2(Ni/Ntot))$$

where Ni is the frequency of phonetic unit i and Ntot is the total number of phonetic units in the utterance set. A user-specified switch determines whether the function is used in its absolute form as given above, or normalized by dividing by the number of letters in the sentences. Most of the later runs were made using the relative form of the function.

Following most authorities on phonetics, we take the relevant set of phonetic units to be phones, allophones or variants of phonemes of American English [4,5], roughly equivalent to Pike's "speech sounds" [6, pp. 42]. The problem of calculating or defining the complete set of allophones is equivalent to defining the set of possible phonological rules. The first-order approximation to this that was used is: an allophone is a variant of a phoneme that is distinguished by the phone on its immediate left, the phone on its immediate right, and, if it is syllabic, by a binary mark of stressed or non-stressed; part of the allophonic representation, also, is whether there are word boundaries on its immediate right or left before the adjacent segments. For the purposes of this specification, left and right environmental phones are the segmental phonemes with vowels marked as stressed or nonstressed and the complex phonemes /ch/, /jh/ written as [t sh] and [d zh]. (This is a correction and generalization of a proposal for psycholinguistic units of speech recognition made by Wickelgren some years ago [7, chap. 6,7].)

It is important to use phones instead of phonemes as possible phonetic conditioning environments for several reasons.

Complex phones condition phonetically according to their separate parts. If you think, as we do, that the vowels of "chew" and "shoe" are phonetically identical, then always counting phones as different if they have different adjacent phonemes won't work: the two vowels have different phonemes on their left -- /ch/ vs. /sh/ -- but the identical phone, [sh]. And /oy/ and /aw/ probably cause rounding assimilation on different ends, /oy/ at the beginning and /aw/ at the end, although there is no principled way to distinguish them with the phonological feature of rounding if they are regarded as holistic segments.

In general, conditioning phones should also be marked redundantly for features that can assimilate over an intervening segment. Only if the /t/ of "stew" is marked for lip rounding will the /s/ be in an environment that will cause it to become rounded, but lip rounding is not phonemic in English consonants. If you think, as we do, that the /s/'s of "stew" and "sty" are phonetically dif-

ferent, then the relevant conditioning environment cannot be just the immediately following phoneme.

Of course, supra-segmental features affect phonetics also. As a first approximation to this, we mark vowels as being stressed or non-stressed and include word and utterance boundaries in conditioning environments. Something like this must be done if you think, as we do, that the /t/'s of "deter" and "veto" are phonetically different, and that the /ay/'s of "Nye trait" and "night rate" are also different.

Because of exigencies of time and resources available, the allophonic codes actually used were 4-byte integers consisting of these bit patterns:

EACH ALLOPHONE CODE:

o 6 bits for segmental phone code
o 6 bits for segmental phone on left
o 6 bits for segmental phone on right
o 1 bit for word boundary on left
o 1 bit for word boundary on right

where segmental phone codes are classical phonemes except:

o Vowels marked stressed/unstressed
o Complex phonemes are split: /CH/=[T SH], /JH/=[D ZH]
o Utterance begin/end mark used: /$/

Figure 3. Allophone Codes.

The simplest way to decode and write one of these allophone codes is as a phone with an environment specified as in linguistic phonological rules. Here is an example from the log of a computer program run testing allophone coding and decoding that shows how this method of counting phonetics handles three well-known phrases that are distinguished by allophones of /T/:

ORTH="Nye trait/nitrate/night-rate"
PRON=/- N AY1 - T R EY1 T - - N AY1
T R EY1 T - - N AY1 T - R EY1 T -/

THESE ARE THE PHONETIC UNITS:

I	ALLO(I)		DECODED:
1	868422		N / [$] __ [AY1]
2	292253		AY1 / [N] __ [# T]
3	643562	*	T / [AY1 #] __ [R]
4	960268		R / [T] __ [EY1]
5	64156		EY1 / [R] __ [T]
6	639957		T / [EY1] __ [# N]
7	878406		N / [T #] __ [AY1]
8	292252		AY1 / [N] __ [T]

```
 9   643560  *  T  / [AY1] __ [R]
10   960268     R  / [T] __ [EY1]
11    64156     EY1 / [R] __ [T]
12   639957     T  / [EY1] __ [# N]
13   878406     N  / [T #] __ [AY1]
14   292252     AY1 / [N] __ [T]
15   643561  *  T  / [AY1] __ [# R]
16   960270     R  / [T #] __ [EY1]
17    64156     EY1 / [R] __ [T]
18   639745     T  / [EY1] __ [$]
```

*: 3 DIFFERENT ALLOPHONES OF /T/.

Figure 4. Allophone Encoding/Decoding

4.5 RESULTING SENTENCES

The sentences resulting from this se-
lection process were checked for
accepability by two experts with Ph.D.'s
in Linguistics with major areas of Phonet-
ics and Phonology (WMF and KGM), and one
Registered Speech Pathologist (Jane
McDaniel), who has been hired as a con-
sultant to help record the data base. The
only area in which there was some
disagreement was on whether or not to
allow utterances consisting of just a
well-formed noun phrase. The decision was
made to allow such fragments if they were
not otherwise unacceptable, because they
are perfectly common and normal in speech,
according to such authorities as Sledd
[8,p. 169]:

> "We often say things, in
> perfectly normal speech, which do
> not contain a complete subject and
> a complete predicate. We might
> very well say, possibly in answer
> to a question,

> the choir ↓

> just as we might say,

> the choir → will sing now ↓

> Both answers are correct English
> utterances, and both end in the
> terminal ."

Figure 5 below shows some character-
istics of the sentences finally selected:

| SENTENCES | #Utts. | Ial | # Allophone | |
			Types	Tokens
DIA+MIT	4410	1584	7,296	~147k
DIA+MIT+TI	6300	2562	19,853	~212k

Figure 5. Phase I Utterances Summary.
Allophone information, Ial, is in kbits.

4.6 SENTENCE-TO-SUBJECT ASSIGNMENT

Sentences were assigned to subjects
represented as indices; as particular real
subjects are chosen they are assigned a
subject index arbitrarily. The initial
assignment of sentences to subject indices
was made by a looping program that assign-
ed consecutive sentences from the selected
sentence set to different subjects.

Another program then re-assigned sen-
tences to subjects in order to reduce the
range of expected total speech durations
assigned to subjects. The program used a
simple fast repetitive heuristic of
finding the subjects with the longest and
the shortest assigned speech durations,
then interchanging the two sentences
between them that make the greatest re-
duction in the difference of their speech
durations, respecting the constraints of
the experimental design (dialect sentences
interchange only with dialect sentences,
MIT only with MIT sentences, and TI only
with TI sentences). Before this program
ran, the minimum, average, and maximum
speech durations assigned to subjects were
23, 30, and 37 seconds respectively;
running the program increased the minimum
to 28.5 and reduced the maximum to 32.

4.7 RECORDING PROGRESS

Last fall TI made a commitment to
send a sample of at least ten speakers'
recordings to NBS for evaluation by
December 21, which was done. In addition,
we made a commitment to record and send
out an average of 20 speakers per week
beginning January 1. Figure 6 below shows
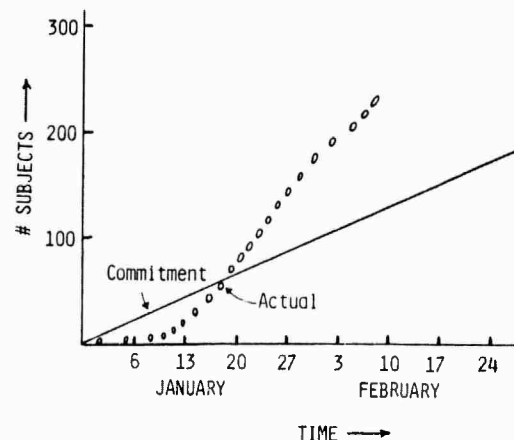how well we have met that commitment.

Figure 6. Recording Progress, Phase I.

We expect to finish Phase 1 and start
Phase 2 recording during April 1986.

Figure 7 below shows the geographical distribution of speakers we have recorded as of February 16, 1986, along with the definition of the assumed dialectal areas we are using in an attempt to get more even numbers of speakers from all major dialects. Table 3 below gives a numerical summary of this same information.

Figure 7. Current Geographical/Dialect Distribution of Speakers.

AREA#	AREA NAME	NMALES	NFEMALES	TOTAL
1	New England	6 (60%)	4 (40%)	10 (4%)
2	Northern	25 (64%)	14 (36%)	39 (17%)
3	North Midland	29 (76%)	9 (24%)	38 (17%)
4	South Midland	34 (67%)	17 (33%)	51 (23%)
5	Southern	25 (63%)	15 (38%)	40 (18%)
6	New York City	4 (57%)	3 (43%)	7 (3%)
7	Western	23 (77%)	7 (23%)	30 (13%)
8	Army Brat	5 (56%)	4 (44%)	9 (4%)
	TOTAL:	151 (67%)	73 (33%)	224

Table 3. Breakdown of Subjects (2/18/86)

4. REFERENCES

[1] "Programs That Design Connected Speech Data Bases", William M. Fisher, JASA 74, supp. 1, p. S48 (A) (Fall 1984)

[2] Computational Analysis of Present-Day American English, Henry Kuchera and W. Nelson Francis, Brown University Press, Providence, Rhode Island, 1967.

[3] Tables of Transitional Frequencies of English Phonemes, Lee S. Hultzen, Josech H.D. Allen Jr., and Murray S. Miron, University of Illinois Press, Urbana, 1964.

[4] A Course in Phonetics, Peter Ladefoged, Harcourt Brace Jovanovich, Inc., New York, 1975.

[5] General Phonetics, R-M. S. Heffner, The University of Wisconsin Press, Madison, 1964.

[6] Phonetics, Kenneth L. Pike, The University of Michigan Press, Ann Arbor, 1966.

[7] Speech and Cortical Functioning, ed. John H. Gilbert, Academic Press, New York, 1972.

[8] A Short Introduction to English Grammar, James Sledd, Scott, Foresman and Company, Chicago, 1959.

SPEECH DATABASE DEVELOPMENT: DESIGN AND ANALYSIS OF THE ACOUSTIC-PHONETIC CORPUS*

Lori F. Lamel, Robert H. Kassel, and Stephanie Seneff

Department of Electrical Engineering and Computer Science, and
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

The need for a comprehensive, standardized speech database is threefold: first, to acquire acoustic-phonetic knowledge for phonetic recognition; second, to provide speech for training recognizers; and third, to provide a common test base for the evaluation of recognizers. There are many factors to consider in corpus design, making it impossible to provide a complete database for all potential users. It is possible, however, to provide an acceptable database that can be extended to meet future needs. After much discussion among several sites, a consensus was reached that the initial acoustic-phonetic corpus should consist of calibration sentences, a set of phonetically compact sentences, and a large number of randomly selected sentences to provide contextual variation. The database design has been a joint effort including MIT, SRI, and TI. This paper describes MIT's role in corpus development and analyzes of the phonetic coverage of the complete database. We also include a description of the phonetic transcription and alignment procedure.

INTRODUCTION

The development of a common speech database is of primary importance for continuous speech recognition efforts. Such a database is needed in order to acquire acoustic-phonetic knowledge, develop acoustic-phonetic classification algorithms, and train and evaluate speech recognizers. The acoustic realization of phonetic segments results from a multitude of factors, including the canonical characteristics of the phoneme, contextual dependencies, and syntactic and extralinguistic factors. A large database will make it possible to examine in detail many of these factors, with the hope of eventually understanding acoustic variability well enough to design robust speech recognizers. A complete database should include different styles of speech, such as isolated words, sentences and paragraphs read aloud, and conversational speech. The speech samples should be gathered from many speakers (at least several hundred) of varying ages, both male and female, with a good representation of the major regional dialects of American English.

DESIGN CONSIDERATIONS

There are many factors to consider in designing a large corpus for speech analysis. Unfortunately, some of the goals are limited by practical considerations. Ideally we would like to include multiple samples of all phonemes in all contexts, a goal that is clearly impractical for a manageable database.

At the last DARPA review meeting it was decided that an initial acoustic-phonetic database would be designed to have good phonetic coverage of American English. It was agreed that the initial acoustic-phonetic corpus would include calibration sentences (spoken by every talker), a small set of phonetically compact sentences (each spoken by several talkers) and a large number of sentences (each to be spoken by a single talker). This combination was chosen to balance the conflicting desires for compact phonetic coverage, contextual diversity, and speaker variability. Another requirement of the corpus was that the sentences should be reasonably short and easy to say.

The database design is a joint effort between MIT, SRI, and TI. The speaker *calibration sentences*, provided by SRI, were designed to incorporate phonemes in contexts where significant dialectical differences are anticipated. They will be spoken by all talkers. The second set of sentences, the *phonetically compact* sentences, was hand-designed by MIT with emphasis on as complete a coverage of phonetic pairs as is practical. Each of these sentences will be spoken by several talkers, in order to provide a feeling for speaker variation. Since it is extremely time-consuming and difficult to create sentences that are both phonetically compact and complete, a third set of *randomly selected* sentences, chosen by TI, provides alternate contexts and multiple occurrences of the same phonetic sequence in different word sequences.

A breakdown of the actual sentence corpus is shown in Table 1. This arrangement was chosen to balance the conflicting desires for capturing inter-speaker variability and providing contextual diversity. Since the calibration

	No. Talkers	No. Sentences	Total
Calibration (SRI)	640	2	1280
Compact (MIT)	7	450	3150
Random (TI)	1	1890	1890
Total	—	—	6320

Table 1: Breakdown of Frequencies of Occurrence of Sentences in Corpus

sentences are spoken by all of the speakers, they should be useful for defining dialectical differences. For multiple instances of the exact same phonetic environments, but with a much richer acoustic-phonetic content than in the calibration sentences, the MIT set would be appropriate. The TI sentences, to be spoken by one talker per sentence, should provide data for phoneme sequences not covered by the MIT database.

DESIGN OF THE COMPACT ACOUSTIC-PHONETIC SENTENCES

A set of 450 sentences was hand-designed at MIT, using an iterative procedure, to be both compact and comprehensive. We made no attempt to phonetically balance the sentences. We used *ALexis* and the Merriam-Webster Pocket Dictionary (Pocket) to interactively create sentences and analyze the resulting corpus. We began with the "summer" corpus created for the MIT speech spectrogram reading course to include basic phonetic coverage and interesting phonetic environments. We initially augmented these sentences by looking at pairs of phonemes, trying to have at least one occurrence of each phoneme pair sequence. *ALexis* was used to search the Pocket dictionary for words having sequences that were not represented and for words beginning or ending with a specific phoneme. We then created sentences using the new words and added them to the corpus. Certain difficult sequences were emphasized, such as vowel-vowel and stop-stop sequences. Some phoneme pairs are impossible; others are extremely rare and occur only across word boundaries. For example, /w/ and /y/ never close a syllable, except as an off-glide to a vowel, so many /w/-phoneme pairs are impossible. After filling some of the gaps in coverage, we reanalyzed the sentences with regard to phoneme pair coverage, consonant sequence coverage, and the potential for applying phonological rules both within words and across word boundaries. In a final pass through the sentence set, we modified and enriched sentences where simple substitutions could introduce variety or generate an instance of a rare phoneme pair.

ANALYSIS OF PHONETIC COVERAGE

This section discusses the phonetic coverage of the compact sentence set developed at MIT and the resulting cor-

pus consisting of the combined MIT and TI sentences. This analysis does not include the calibration sentences as we consider their use to be of a different nature.

	POCKET	HL	MIT-450	APDB
# sentences		720	450	5040
# unique words	19,837	1894	1792	5107
# words	19,837	5745	3403	41,161
ave # words/sent		7.9	7.6	8.2
min # words/sent		5	4	2
max # words/sent		12	13	19
ave # syls/word	1.38*	1.1	1.58	1.54
ave # phones/word	3.34*	2.97	4.0	3.89

* The ave # syls/word and ave # phones/word have been weighted by Brown Corpus[1] word frequencies.

Table 2: Description of Databases

Table 2 compares some of the distributional properties of the Pocket Lexicon (Pocket), the Harvard List (HL)[2], the MIT-selected sentences (MIT-450), and the Acoustic-Phonetic Database selected sentences (APDB). The APDB includes seven copies of each MIT-450 sentence, to account for the number of talkers per sentence, and a single copy of each randomly selected sentence (TI-1890). Since we were given only the orthographies for the TI-1890 sentences, we generated phonemic transcriptions by dictionary lookup, by rule-based expansion of the dictionary entries, and, as a last resort, by a text-to-speech synthesizer. We expect that there are pronunciation variations between the dictionary and the text-to-speech synthesizer, particularly with respect to vowel color. There may also be some pronunciation errors, but we think these will be statistically insignificant.

The proportion of unique words relative to the total number of words is substantially larger in the MIT-450 than the APDB, probably due to the selection procedure. We tried to use new words in sentences and to avoid duplication when at all possible. Roughly 50% of the MIT-450 words are unique, as compared to only 25% of the APDB words. The TI-1890 sentences are, on the average, slightly longer than those in the MIT-450. The 10 most frequently occurring words for all of the corpora are function words or pronouns. In both the MIT-450 and the APDB corpora, the most common word is "the," accounting for roughly 7% of all words.

The average numbers of syllables and phones per word are longer for the MIT-450 and the APDB than for the HL. This is presumably due to the higher percentage of polysyllabic words.

Figure 1 shows the distribution of the number of syllables per word for the two corpora. The distributions are quite similar, with the majority of the words being mono- or bi-syllabic. The MIT-450 corpus has a slightly higher percentage of polysyllabic words than does the combined
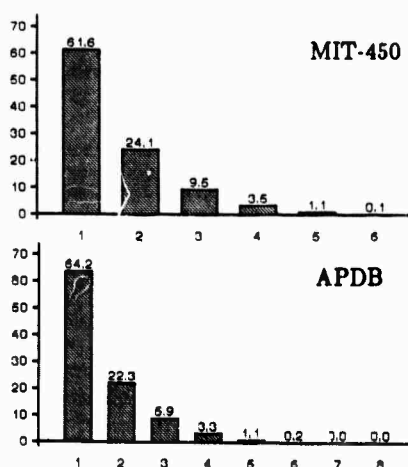
Figure 1: Histograms of the number of syllables per word.

Figure 3: Histograms of the 10 most common phonemes.

corpus. We specifically tried to include polysyllabic words in the sentences, since these are likely to be spoken with greater variability.

Distributions of the number of phonemes per word are shown in Figure 2. The 10 most common phonemes and their frequency of occurrence are given in Figure 3.

Table 3 shows the distribution of within-word consonant sequences for the four databases. The MIT-450 sentence set covers most of the consonant sequences occurring within words. The APDB has more complete coverage, particularly for the word-final and word-medial sequences. We examined a list of all of the word-initial and word-final clusters in the sentence list, and compared these with the occurrences in Pocket. We verified that essentially every initial cluster that occurred more than once in the Pocket lexicon was included at least once in the APDB, and that most of the final clusters were covered. Often, if a word-final cluster did not occur in word-final position in the APDB, the sequence did occur within a word or across a word boundary. Generally, the sequences occurring in Pocket that are not covered by APDB are from borrowed words such as "moire" and "svelte."

The APDB includes many word-final consonant sequen-

	POCKET	HL	MIT-450	APDB
# unique words	19,837	1894	1792	6103
# WI	75	59	64	68
# WF	129	105	102	146
# WM	608	123	228	388
# boundaries		4305	2953	36,121
# WB		976	805	1639

Table 3: Distribution of Consonant Sequences

ces that were not present in MIT-450. In fact, there are more word-final consonant sequences in the APDB than actually occur in Pocket. The reason is that the Pocket lexicon does not include suffixes.

A more detailed phonetic analysis of all *phoneme pairs* is included in Appendix 1 in tabular form. The tables are broken down into phoneme subsets, and data are included for both the MIT-450 and the APDB. Some of the gaps in the MIT-450 table have been filled in by sentences in the TI-1890 corpus (e.g., the syllabic /l/ column of the vowel-sonorant pairs table and the /y/ column of the vowel-sonorant pairs table). Note also that some gaps occur in both tables. Such gaps are expected, since some phoneme sequences are impossible or quite rare. For example, the lax vowels (excluding schwa) are never found in syllable-final position in English. As a consequence, table entries requiring lax vowels as the first member of a pair have many gaps (see for example, the vowel-vowel entries in the pair tables.)

Figure 4 compares histograms of the sentence types for the MIT-450 and the APDB. Simple sentences (Simple S.) and questions (Simple Q.) have no major syntactic markers. Complex sentences (Complex S.) and questions (Complex Q.) are expected to have a major syntactic boundary when read. As can be seen, the APDB has a wider variety of sentence types, with 75% being simple declarative sentences. In the MIT-450, almost 85% of the sentences are of the simple declarative form. Questions form about

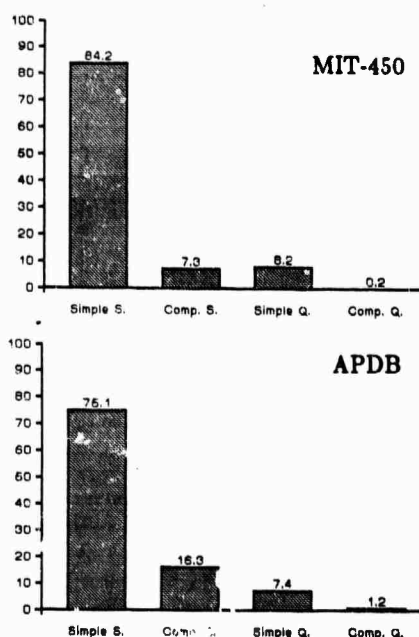**Figure 2:** Histograms of the number of phonemes per word.

Figure 4: Histogram of sentence types.

10% of both corpora.

Figure 5 shows counts of environments where major phonological rules may apply. We chose to gather information on the following possibilities:

- gemination (GEM)
- vowel-vowel sequences (VVS)

- vowel-schwa sequences (VSS)
- schwa-vowel sequences (SVS)
- flapping of /t/,/d/, and /n/ (FLAP)
- homorganic stop insertion (HSI)
- schwa devoicing (S-DVC)
- fricative devoicing (F-DVC)
- /s/-/š/ and /z/-/ž/ palatalization (PAL)
- y-palatalization: /dy/→/ǰ/ (DY-Jh)
- y-palatalization: /ty/→/č/ (TY-Ch)
- y-palatalization: /sy/→/š/ (SY-Sh)

The histograms show that both corpora have many potential environments for flapping and homorganic stop insertion. The vowel-vowel environments are also well covered. The analysis for phonological rule application is difficult, because of the difficulties in predicting what different speakers will say.

RECORDING, LABELING, AND ALIGNMENT

The recording of the sentences is currently under way at TI. Speech is recorded digitally at 20 kHz, simultaneously on a pressure-sensitive microphone and on a Sennheiser close-talking microphone. Digital tapes are shipped to NBS, where they are filtered and downsampled to 16 kHz. The resampled tapes are then shipped to MIT where the orthographic and phonetic transcriptions are generated.

Transcriptions are generated using the *Spire* facility, in conjunction with the automatic alignment system pro-
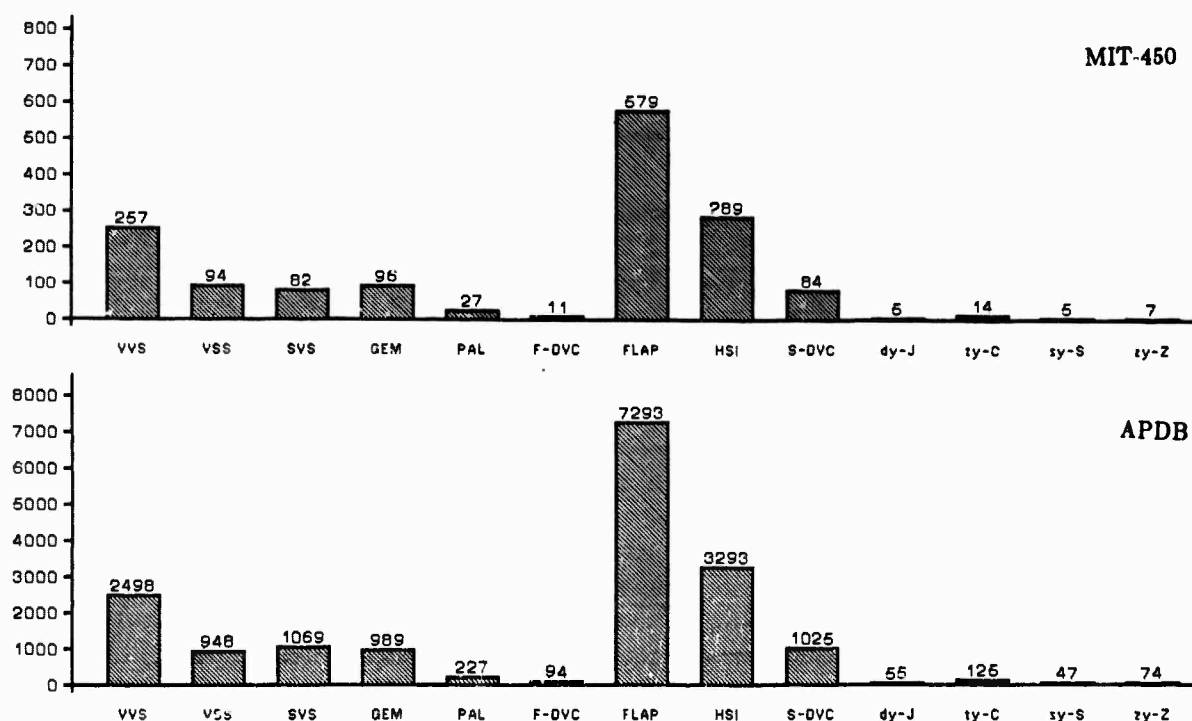
Figure 5: Histogram for potential application of phonological rules.

103

Unvoiced Stops:	p t k č
Voiced Stops:	b d g ǰ
Stop Gaps:	pᵒ tᵒ kᵒ ʔ bᵒ dᵒ gᵒ ɾ
Nasals:	n m ŋ ĩ
Syllabic Nasals:	ņ m̧ ŋ̧ ļ
Unvoiced Fricatives:	s š f θ
Voiced Fricatives:	z ž ⌄ ð
Glides:	l r w y
Vowels:	ɪʳ ɪ ɛ eʸ æ ɑ ɑʷ ɑʸ
	ʌ ɔ ɔʸ ɑʷ ʊ u ū ʒ
Schwa:	ə ə̧ ɪ ɘ
H, Silences:	h ɦ ◫ ◻

Figure 6: Phones used for labeling.

vided by Leung [3]. The transcription process involves three steps:

1. A "Phonetic Sequence," which consists of a list of the phones of the utterance in correct temporal order but with no boundaries marked in time, is entered.

2. The utterance is run through an automatic system to generate an alignment for the sequence.

3. The automatically generated alignment is hand-corrected.

Only the data recorded through the pressure microphone are transcribed. Transcriptions for the close-talking version are generated by duplicating the results for the pressure microphone.

The phones used in the labeling are shown in Figure 6. In many cases, it is not possible to define a boundary between two phones, such as /ɔr/, because features appropriate for both phones often occur simultaneously in time. When no obvious positioning of the boundary is apparent, arbitrary rules, such as an automatic 2/3:1/3 split, are invoked. There are also some cases in which none of our standard phones are appropriate for a given portion of the speech, primarily because of severe coarticulation effects. In such cases, the segment is labeled as the nearest phone equivalent, according to the transcriber's judgment. There are other difficult cases, such as syllable-initial /pl/, where the /l/ is devoiced at onset. Should the portion before voicing begins be thought of as part of the aspiration of the /p/, or as part of the /l/? We have decided, somewhat arbitrarily, to define the onset time of the phone following an unvoiced stop as coincident with the onset of voicing. These remarks serve simply as examples of some of the difficulties that arise in transcribing continuous speech. We are mainly interested in using consistent methods of transcribing in situations where ambiguity exists. Currently the transcription rate is 100 sentences per week.

SUMMARY

We have described various components of the preliminary acoustic-phonetic database and discussed some of the issues in its design. Evaluating the phonetic coverage of the database is difficult primarily because no

dard for comparison exists. We have chosen to compare the phonetic coverage of the database to two well-known sources, the Merriam-Webster Pocket Dictionary of 1964 and the Harvard List sentences. The dictionary does not reflect spoken English very well, and can only guide us in judging the possible phonemic sequences within words. The Harvard List sentences, while phonemically balanced, consist primarily of very simplistic sentences and monosyllabic words. In addition, they are balanced for phoneme occurrences, whereas we tried to account for occurrences of phoneme pairs.

We believe that we have adequate coverage of most phonemes and phoneme pairs. In cases where the phoneme pairs are scarce, there are often other phoneme pairs that will provide similar information. For example, the class sequence [alveolar consonant] [back vowel] is more general than /t/ /ɔ/, and has a higher frequency of occurrence.

We hope that the APDB database will provide guidelines for the development of future databases. An analysis of the spoken corpus will enable us to judge our phonetic analysis procedure. In particular, we will be able to evaluate the relationship between our phonological rule predictions and the frequency with which a phonological modification actually occurred.

REFERENCES

[1] Kucera, H. and W.N. Francis (1967) *Computational Analysis of Present-Day American English*, Brown University Press, Providence, R.I.

[2] Egan,J. (1944) "Articulation testing methods II," OSRD Report No. 3802, U.S. Dept. of Commerce Report PB 22848, November.

[3] Leung, H. C. and V.W. Zue (1984) "A Procedure for Automatic Alignment of Phonetic Transcriptions with Continuous Speech," *Proc. ICASSP-84*, 2.7.1-2.7.4.

Appendix 1

MIT

	iʸ	eʸ	ɔ	oʷ	u	aʸ	ɔʸ	aʷ	ɝ
iʸ	3	7	8	9		3	1	3	1
eʸ	1	1	4	4		1		1	1
ɔ	1	1	1	1				1	
oʷ	2	2	3	1		1	1	3	
u	1	5	5	3	1	2	1	2	1
aʸ	4	4	6	2	1	3	1		1
ɔʸ	1	1	1	1					
aʷ	1	2	1	1		1	1		
ɝ	7	1	1	2					1
ɪ									
ɛ									
æ									
a	1								
ʌ									
ʊ									
ɚ	4	7	4	1		6	1	3	1
ə	3	1		7		4	1	1	1
ɨ	1								

APDB

	iʸ	eʸ	ɔ	oʷ	u	aʸ	ɔʸ	aʷ	ɝ
iʸ	28	70	75	88		27	7	26	22
eʸ	10	7	38	31		10		7	9
ɔ	7	7	7	7				7	
oʷ	14	14	26	10		9	7	21	4
u	12	48	43	26	7	19	7	16	10
aʸ	28	30	47	16	8	21	7	2	8
ɔʸ	10	7	7	7					
aʷ	7	14	9	7		7	7		
ɝ	62	10	12	18		3		2	7
ɪ									
ɛ			2						3
æ									
a	8								
ʌ									
ʊ									
ɚ	64	73	37	13		52	10	25	7
ə	22	3	7	60		33	7	11	10
ɨ	7								

	iʸ	eʸ	ɔ	oʷ	u	aʸ	ɔʸ	aʷ	ɝ
n	38	5	8	11	10	14	4	3	
m	17	24	10	18	9	24	2	1	4
ŋ	1	1	3					1	1
m̩			1						
n̩		1							
l̩	2		2			1		1	
l	88	23	13	25	9	22	4	12	7
r	81	28	16	28	24	24	3	5	
w	20	20	20	6	1	17	1	1	10
y			1	1	88				1

	iʸ	eʸ	ɔ	oʷ	u	aʸ	ɔʸ	aʷ	ɝ
n	422	79	88	252	117	133	32	95	21
m	246	318	100	227	86	248	18	22	48
ŋ	9	7	32	8		1	1	10	7
m̩			7						
n̩		7							
l̩	55	5	17	5		13		7	2
l	1001	287	168	286	115	308	34	110	60
r	936	312	169	303	260	272	29	71	6
w	294	212	199	67	8	210	7	9	134
y	3		9	7	933			5	10

	iʸ	eʸ	ɔ	oʷ	u	aʸ	ɔʸ	aʷ	ɝ
b	27	2	9	10	1	28	4	3	8
d	18	22	8	10	14	7		9	2
g	1	3	1	13	4	1		1	1
p	14	14	2	11	2	9	8	3	12
t	44	19	11	13	86	18	1		4
k	9	9	19	15	7	3	4	8	11
č	6	9	1	2	4	3	1		3
ǰ	7	2	3	1	6	2	5		4
s	22	5	12	5	9	13	2	2	10
z	14	7	10	3	2	2		1	4
š	11	2	5	5	3	1		1	1
ž					1				
f	9	5	17	46	2	8		3	12
v	10	6	2	5		3	2	1	4
θ	6		2					3	5
ð	3	9		5					
h	8		7	4	5	10		12	18

	iʸ	eʸ	ɔ	oʷ	u	aʸ	ɔʸ	aʷ	ɝ
b	364	35	78	118	15	282	48	54	79
d	270	212	95	133	165	80	1	107	44
g	7	51	18	137	29	15	1	7	28
p	149	141	32	142	21	81	75	38	137
t	501	234	157	175	1060	220	9	21	99
k	99	137	191	146	56	31	32	75	106
č	54	73	10	18	56	28	9	1	26
ǰ	61	16	25	12	55	17	40		36
s	275	83	137	176	85	181	20	26	123
z	130	63	109	58	18	33		12	37
š	198	33	42	54	35	9		11	15
ž					22				
f	120	72	172	506	25	117	4	32	120
v	92	68	28	44		60	28	15	63
θ	56	2	31	3	3	3		37	53
ð	65	192		59				2	3
h	381	18	58	79	54	117		137	213

MIT

	I	ε	æ	ɑ	ʌ	ʊ	ɝ	ə	i
iʸ	11	4	16	11				5	28	1
eʸ	4	3	3	1	1			1	3	
ɔ	1	1	1	2						1
oʷ	4	1	2	3				1		3
ʊ	6	7	4	6	1			1	4	
ɑʸ	5	2	7	3	2			4	11	6
ɔʸ	3	1	1					2	1	2
ɑʷ		1	1		1			3	3	
ɝ	2		1		1				1	1
I										
ε										
æ										
ɑ		.								
ʌ										
ʊ										
ɝ	6	4	7	6	2			1	7	4
ə	5	6	8	2	2				3	
i										

APDB

	I	ε	æ	ɑ	ʌ	ʊ	ɝ	ə	i
iʸ	174	47	182	102	7		46	399	27
eʸ	38	22	26	21	9		8	38	7
ɔ	9	7	9	14	2			3	8
oʷ	54	14	26	25	3		28	28	34
ʊ	75	58	48	54	11		8	79	13
ɑʸ	51	18	61	23	14		30	107	61
ɔʸ	21	8	8				14	7	15
ɑʷ	5	17	9	1	8		32	30	
ɝ	27	8	13	1	9			30	16
I									
ε								1	
æ									
ɑ									
ʌ									
ʊ									
ɝ	89	38	73	48	21		8	106	52
ə	57	52	82	19	23			42	2
i									

	I	ε	æ	ɑ	ʌ	ʊ	ɝ	ə	i
n	38	16	13	19	10		16	29	14
m	17	15	13	7	23		9	44	8
ŋ	12		3	1				2	2
m̩									
ṇ	2	3	1	1				1	1
ḷ	10	2	2	1				7	3
l	37	22	29	20	11	5	12	33	24
r	69	39	33	26	38	1	2	49	16
w	60	25	3	7	7	12	6	30	
y	4	4		3	6	38	2	1	1

	I	ε	æ	ɑ	ʌ	ʊ	ɝ	ə	i
n	434	185	185	260	148	3	144	388	214
m	231	223	219	93	240		91	516	87
ŋ	125	6	51	10	11			43	14
m̩	1		3					1	
ṇ	18	24	11	9			2	11	10
ḷ	93	29	37	10	2			96	43
l	397	281	330	191	127	67	119	342	279
r	808	415	364	271	393	10	17	562	211
w	659	293	31	125	150	145	144	370	5
y	44	55	3	29	46	353	44	15	9

	I	ε	æ	ɑ	ʌ	ʊ	ɝ	ə	i
b	25	12	16	15	11	3	5	12	5
d	56	9	17	6	7	1	16	56	22
g	9	10	9	13	7	4	9	3	5
p	12	13	14	20	2	5	23	16	3
t	63	21	30	16	8	2	44	47	45
k	10	11	42	25	23	6	6	33	14
č	6	4	2	7	1		10	3	6
ǰ	12	12	4	4	6		5	2	25
s	44	15	11	8	19	1	7	39	20
z	37	7	17	13	4		4	30	9
š	5	5	3	2		9	1	3	42
ž						1	4		3
f	23	5	6	5	5	2	2	15	2
v	12	12	10	3			33	16	3
θ	14	1	1	1			4	3	
ð	15	14	21		1		12	219	
h	24	7	31	15	3	3			

	I	ε	æ	ɑ	ʌ	ʊ	ɝ	ə	i
b	271	123	185	145	204	31	69	185	41
d	646	130	209	82	121	10	174	595	245
g	99	132	97	118	67	57	87	30	54
p	169	150	166	208	41	65	222	223	44
t	764	288	326	160	105	35	456	690	481
k	127	97	415	266	251	81	51	404	178
č	61	41	29	56	14	4	101	47	70
ǰ	110	120	35	39	68	1	50	29	230
s	523	349	166	88	268	8	86	488	270
z	386	69	228	157	52		36	437	112
š	56	48	40	32	14	112	13	84	172
ž			1			7	39	3	34
f	268	87	96	74	46	26	39	172	31
v	152	145	124	40	11		345	230	52
θ	171	16	18	7	3		34	50	19
ð	238	196	323			14	187	2230	
h	455	118	452	137	46	27	9	2	4

MIT

	b	d	g	p	t	k	č	ǰ
iʸ	7	27	9	27	32	25	17	1
eʸ	8	13	2	9	47	17	2	7
ɔ	2	5	5	1	13	4	3	1
oʷ	5	2	2	9	15	12	3	
u	10	14	5	15	17	10	4	7
aʸ	7	17	2	2	24	14	1	1
ɔʸ	2	22	1	2				
aʷ		2	1	1	22	1		
ʒ	8	5	1	5	10	10	6	5
I	5	13	28	16	21	87	8	12
ɛ	2	16	8	5	20	37	1	8
æ	10	10	10	9	37	27	4	2
a	13	7	3	19	31	19	2	10
ʌ	10	3	5	15	12	4	8	3
U	1	24	1	1	4	13	1	
ɚ	6	14	2	3	10	8	4	
ɔ	48	46	28	45	43	46	2	14
i	8	13	2	3	27	9		8

APDB

	b	d	g	p	t	k	č	ǰ
iʸ	116	357	102	292	387	270	155	13
eʸ	113	188	20	84	503	228	18	62
ɔ	16	41	48	8	130	52	23	7
oʷ	68	31	30	104	157	124	27	
u	121	173	58	152	196	120	28	57
aʸ	60	249	19	31	292	166	7	8
ɔʸ	15	22	7	14	1	3		
aʷ	3	42	9	8	262	8		3
ʒ	69	75	11	53	121	112	52	46
I	48	186	266	186	501	873	100	145
ɛ	16	224	76	83	269	438	8	71
æ	95	192	99	118	519	357	43	26
a	133	91	33	195	408	183	16	87
ʌ	101	54	60	171	202	64	104	23
U	8	314	10	8	48	149	7	
ɚ	63	203	23	43	101	83	34	5
ɔ	537	616	307	477	517	514	31	116
i	79	168	28	37	387	137		75

	b	d	g	p	t	k	č	ǰ
n	13	119	7	11	120	17	5	16
m	15	3	3	33	6	6	1	1
ŋ	3	4	9	3	4	10	1	1
m̥								
n̥	1	3			11		1	
l		5	4	5	8	4	1	1
ḷ	7	29	1	6	7	8	1	1
r	16	36	9	5	22	15	5	15
w								
y								

	b	d	g	p	t	k	č	ǰ
n	144	1482	72	118	1343	204	76	155
m	181	63	26	369	80	55	9	9
ŋ	39	37	125	41	77	146	7	10
m̥		1						
n̥	8	28		1	141	2	7	
l	24	75	40	55	84	47	10	1
ḷ	71	348	15	74	129	79	14	15
r	156	359	93	86	277	173	41	121
w								
y								

	b	d	g	p	t	k	č	ǰ
b	3	5	1	2	4	2	1	3
d	25	5	3	5	14	8	2	3
g	4	3	3	2	3	1	1	1
p	4	1	2	4	11	2	2	1
t	18	14	6	11	18	13	1	3
k	2	6	3	7	45	6	1	1
č	2	2	3	1	4	3	2	1
ǰ	1	8	1	3	4	1	2	2
s	9	7	4	46	158	56	7	1
z	15	21	9	11	16	17	2	4
š	1	1	1	1	4	1	1	
ž		1						
f	1	1	1	2	12	3	1	
v	1	3	1	7	3	5	1	1
θ		5		2	1	2		1
ð	2	1		1		1	1	
h								

	b	d	g	p	t	k	č	ǰ
b	24	41	8	14	30	14	7	27
d	283	82	51	95	196	105	16	30
g	28	30	21	17	21	8	7	7
p	33	11	16	32	138	18	15	7
t	210	161	75	119	214	154	15	24
k	26	48	31	58	513	59	26	7
č	16	17	22	9	46	26	14	8
ǰ	12	67	8	25	31	9	14	14
s	98	77	39	503	1778	523	69	12
z	174	221	85	124	189	175	19	40
š	8	8	7	8	44	7	7	
ž		7						
f	13	17	8	15	155	30		7
v	36	77	21	67	50	61	8	9
θ	7	38	3	16	14	24		7
ð	14	8		7		7	7	
h								

107

MIT

	n	m	ŋ	m̥	n̥	l̥	l	r	w	y
iʸ	23	15				1	15	11	15	2
eʸ	29	11					8	1	2	1
ɔ	29	1	10				39	61		
oʷ	24	6					20	77	3	
u	15	23				1	9	5	7	5
aʸ	16	15				1	15	12	2	2
ɔʸ	9	1					4			
aʷ	20	2					2	10	2	
ɝ	8	7					4	1	5	
ɪ	112	32	46				49	24	2	
ɛ	56	17	3				48	26		
æ	125	24	8				23	31		
a	17	15	1				16	100		
ʌ	46	34	11				11			
ʊ		2					16	30		
ɚ	7	5				5	6	2	13	2
ə	104	62					50	10	10	4
ɨ	94	1	60							

	n	m	ŋ	m̥	n̥	l̥	l	r	w	y
n	5	15				8	9	3	8	8
m	3	10				2	5	2	3	10
ŋ	2	1					2	5	5	4
m̥							1			
n̥						2		1		
l̥	2	1					4	3	2	
l	2	8					3	3	12	8
r	16	23			1	2	18	5	13	6
w							1			
y						1				

	n	m	ŋ	m̥	n̥	l̥	l	r	w	y
b	1	1				21	21	27	2	9
d	8	5			11	4	7	32	10	5
g	3	2				4	8	36	6	5
p	4	3				10	41	57	4	16
t	6	10		5	5	7	18	63	16	14
k	2	3			1	25	36	30	26	18
č	1	1			1	1	2	1	1	2
ǰ	1	2					1	1	1	1
s	5	18			6	2	16	4	18	5
z	9	14		2	7		4	9	14	7
š	1	3			7	5	2	2		1
ž		1								
f	1	1				6	12	35	1	6
v	8	5			1	6	3	12	3	5
θ	1	1					1	12	1	
ð				1						1
h									18	5

APDB

	n	m	ŋ	m̥	n̥	l̥	l	r	w	y
iʸ	276	224				41	213	143	219	31
eʸ	293	157				6	109	32	43	10
ɔ	335	10	127				480	574		2
oʷ	293	92				1	258	987	38	3
u	161	247				60	115	72	78	57
aʸ	220	171				14	161	151	22	17
ɔʸ	83	7				5	55	1		2
aʷ	241	22				2	21	122	20	2
ɝ	104	79				6	63	19	43	6
ɪ	1372	423	500				509	330	16	
ɛ	795	213	27				540	340		
æ	1544	233	90			1	231	350		
a	201	160	8				183	1022		
ʌ	522	437	108				121			
ʊ		28					163	299		
ɚ	74	58				63	62	35	110	20
ə	1324	680	6				537	162	152	37
ɨ	1013	37	758				5			

	n	m	ŋ	m̥	n̥	l̥	l	r	w	y
n	59	169				88	129	50	113	108
m	33	88				33	49	24	53	90
ŋ	23	28					33	48	50	31
m̥	1						7		2	
n̥	1	1			20		7	1	9	
l̥	24	20					85	31	24	2
l	27	85		2			29	41	115	81
r	160	262			9	34	167	55	142	58
w						8		1		
y					7		1			

	n	m	ŋ	m̥	n̥	l̥	l	r	w	y
b	8	9				236	228	274	17	82
d	114	106			116	43	123	330	148	55
g	46	19				43	98	357	47	56
p	36	33				102	425	633	35	85
t	99	152		39	60	102	222	728	250	125
k	36	37			7	251	376	324	287	185
č	9	12			7	15	20	10	13	14
ǰ	8	16			4		10	7	9	11
s	66	184			70	31	197	54	213	47
z	154	163		29	100	8	71	98	175	74
š	10	24			49	55	17	22	1	7
ž		7							1	
f	10	11				98	149	330	16	71
v	68	71			8	74	44	129	37	54
θ	9	14		1		1	16	128	15	1
ð		1		7						7
h							1		315	44

MIT

	s	z	š	ž	f	v	θ	ð	h
i^y	28	43	3	1	11	18	2	6	6
e^y	14	12	20	2	2	5	1		1
ɔ	9	6	2		12		5		2
o^w	14	18	5	2	2	16	4	1	2
u	18	25	3		8	9	5	10	8
a^y	15	17	1		2	8		4	2
$ɔ^y$	7	5						1	
a^w	6	1							
ɝ	10	7	1		5	6	4	2	1
I	50	64	32	2	21	22	7	24	
ɛ	27	3	7	2	4	25		3	
æ	17	15	6	1	17	15	3	1	
a	11	1	2	5	2	2			
ʌ	16	2	3		4	7	2	7	
U									
ɚ	7	36	2		9	4	1	7	8
ə	67	54	3		28	55	6		14
ɨ	43	17			2	2	1		

APDB

	s	z	š	ž	f	v	θ	ð	h
i^y	374	435	52	9	170	194	56	102	158
e^y	194	121	248	15	36	72	13	10	22
ɔ	105	55	19		147	1	45	5	14
o^w	174	184	59	17	24	165	51	29	33
u	193	216	38	19	88	91	53	131	109
a^y	154	211	11		53	94	3	42	23
$ɔ^y$	67	45			2			7	
a^w	55	22	2		2	5	7	7	7
ɝ	129	97	17		51	68	44	23	13
I	647	901	290	21	252	251	152	168	5
ɛ	332	43	67	23	47	286	12	33	1
æ	206	263	67	9	166	180	28	14	1
a	119	11	16	36	18	19	2	8	1
ʌ	238	52	30		58	83	25	111	
U		2	5		1				
ɚ	100	346	23		86	40	10	83	95
ə	797	639	66		325	825	66	2	174
ɨ	452	162	12		34	46	7		1

	s	z	š	ž	f	v	θ	ð	h
n	60	45	7		12	1	3	34	11
m	6	20	1		8	1	2	11	4
ŋ	10	5	2		5	2	2	4	8
m̩		2							
n̩	2	4			1	1		1	
l̩	6	17			6	1	1	1	
l	7	13	1		11	3	5	5	4
r	15	16	2		8	3	2	9	1
w									
y									

	s	z	š	ž	f	v	θ	ð	h
n	718	451	83		165	43	37	367	180
m	93	213	9		81	9	26	97	58
ŋ	94	61	17		48	18	23	54	82
m̩		18	1					1	
n̩	25	36			9	7	1	10	2
l̩	64	169	4		67	12	10	11	24
l	123	152	16		140	52	39	57	58
r	191	157	25	1	101	30	27	110	60
w									5
y									

	s	z	š	ž	f	v	θ	ð	h
b	2	4	1		1	1	1	1	1
d	15	17	1		15	3	4	15	11
g	1	11	1	1	1		1	1	1
p	13	1			2	1	1	3	1
t	77	1	2		12	2	1	21	12
k	53	1	4		6	1	1	4	2
č	1		1		1		1	2	1
ǰ	7	1	7		10	1	2	7	7
s									
z	19		9		19	2	2	5	16
š	3	1	1		3	1	1		
ž	1								1
f	6				1	1	3	1	
v	4	9	1		6	1		11	5
θ	3	1			2	1	1		
ð	1				2	1		4	1
h									

	s	z	š	ž	f	v	θ	ð	h
b	32	36	7		8	12	7	7	8
d	191	233	25		174	54	43	191	214
g	10	131	8	7	10		8	8	10
p	152	7	11		23	7	7	35	14
t	910	7	41		158	24	18	272	233
k	604	7	81		66	11	8	49	47
č	32	7	7		16	7	7	14	14
ǰ	11		7		9		7	16	11
s	92	7	61		127	14	21	95	107
z	227	1	80		185	26	22	129	204
š	22	7	7		22	7	9	1	3
ž	7							1	7
f	62		7		10	7	21	17	14
v	72	98	10		58	13	2	152	71
θ	35	7	2		20	9	8	16	15
ð	7	3			15	7		30	7
h									

THE DEVELOPMENT OF SPEECH RESEARCH TOOLS ON MIT'S LISP MACHINE-BASED WORKSTATIONS*

D. Scott Cyphers, Robert H. Kassel, David H. Kaufman,
Hong C. Leung, Mark A. Randolph, Stephanie Seneff,
John E. Unverferth, III, Timothy Wilson, and Victor W. Zue

Research Laboratory of Electronics, and
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

In recent years, a number of useful speech- and language-related research tools have been under development at MIT. These tools are aids for efficiently analyzing the acoustic characteristics of speech and the phonological properties of a language. They are playing a valuable role in our own research, as well as in research conducted elsewhere. This paper describes several of the systems being developed for use on our Lisp Machine workstations.

INTRODUCTION

In many areas of speech research, ranging from speech analysis to synthesis to recognition, researchers often follow a common set of analysis procedures. Specifically, there is frequently a need to:

- record and digitize utterances, and define and compute various attributes of the speech signal,

- display and perform interactive measurements of these attributes,

- obtain statistical descriptions of the interrelation between acoustic and phonetic events by examining a large speech database,

- investigate the phonological properties of the language at the symbolic level, using large lexicons and/or printed text, and

- interactively synthesize speech in order to study the relative merits of acoustic cues for phonetic contrasts.

The ability to perform these tasks with ease will greatly facilitate the gathering of information and the corresponding improvement of our speech knowledge. One of our ongoing activities at MIT is the development of a speech research tools to satisfy these needs. This facility has already proven so useful that the necessary hardware and software have been acquired by many laboratories around the world. This paper is intended to provide a progress

report on the development of these tools. It begins with a discussion of hardware requirements and then focuses on software tools.

HARDWARE REQUIREMENTS

The speech workstation that we are refining centers around a Symbolics 3600 series Lisp Machine with 4 Mbytes of main memory, a 474 Mbyte disk, a floating-point accelerator, and a FEP or Generic-Bus Unibus interface. The Lisp Machine's high-resolution graphics console and mouse provide extremely convenient user interfaces.

The Lisp Machine may be augmented with a Floating Point Systems FPS-100E or FPS-5100 array processor for speed-up in numerical computations, and with a Digital Sound Corporation DSC-200/240 A/D and D/A converter for data input/output. The workstation also has a shared Versatec V-80 electrostatic printer/plotter, and assorted audio equipment such as a microphone, a set of headphones, and a tape recorder. The Lisp Machine workstations are connected to one another and to central file servers via a packet-switched local area network.

SOFTWARE SYSTEMS

Several interactive speech research tools are under development on the Lisp Machine workstation. In this section we will describe four such systems: *Spire*, *Search*, *ALexiS*, and *Synth*.

Spire

Spire (Speech and Phonetics Interactive Research Environment) is a software package that enables users to digitize, transcribe, process, and examine speech signals. A variety of different computations can be performed on the signal, and the results can be conveniently displayed and measured.

Spire organizes an utterance as a collection of *attributes*. The attributes may be either symbolic (e.g., phonetic transcription) or numeric (e.g., RMS amplitude). Some of the attributes are one-dimensional (e.g., speech waveform), while others are multidimensional (e.g., a series of short-time spectra). *Spire* has knowledge of the properties and

parameters of the attributes. As a result it is convenient to redefine parameter values (such as LPC order) and to define an attribute that depends upon another attribute.

Displays in *Spire* are organized in the form of *layouts*. The recording and transcription layouts are provided by *Spire*, since these are almost always needed by users. Other frequently used layouts can also be predefined. Many of the commands in *Spire* are given with the hand-held mouse pointer. The mouse can be used to configure a layout, play a section of the utterance, edit waveforms, examine data values, alter display options, and perform other functions.

Spire has been designed with two general goals in mind. First, it is intended to provide an extremely interactive environment and a basic set of capabilities such that speech scientists, even those with little or no programming experience, are able to collect and analyze speech data. Second, *Spire* is designed for easy customizing: users can readily add new attributes to suit their research needs. Currently the core of *Spire* defines default computations for approximately 40 attributes of the speech signal, whereas a customized version can define any number of attributes. Some of the customized *Spire* systems in our research group have as many as 300 attributes. The remainder of this section gives some examples of the operation and capabilities of *Spire*. For a detailed description of *Spire*, see Cyphers [2].

Collecting Speech Speech can be sampled at any rate up to 75 kHz, with appropriate anti-aliasing filters selected by the user. Up to 60 seconds of speech (the maximum is controlled by a parameter) can be digitized at a time. Currently the speech samples are transferred to virtual memory. In the near future, speech will be transferred directly to disk so as to permit the digitization of much longer samples of speech. An automatic end-point detector attempts to locate each utterance. The user can listen to the located utterance, modify the endpoints, and accept the utterance into the database, all with several clicks of a mouse button.

Information about the talker, sampling rate, file name, and orthographic transcription can be changed easily with a click of the mouse. Alternatively, an agenda file can be set up to sequentially change these parameters automatically. This latter option is particularly useful for bulk data input when a list of the recorded utterances already exists on-line.

Signal Processing Since most signal processing is computationally expensive, *Spire* provides mechanisms for reducing unnecessary processing. One way this is done is by ensuring that nothing is computed until it is needed. Once something has been computed, *Spire* remembers the value for the duration of the user's session, and will reuse that value if it is needed again, unless the user specifically forces it to be recomputed. All of this happens in a way that is virtually transparent to the user. Whenever

the amount of overhead involved in data transfer is justified, computations are done on the FPS. Most of the basic analysis procedures are also written to run (albeit more slowly) in Lisp when the machine is without an FPS.

Attributes may also be saved in parameter files and reloaded during future sessions. When there are multiple parameter files associated with a particular utterance, *Spire* reassociates the parameters with the appropriate utterance as they are loaded. Several attributes can be precomputed on a large database (which may take many hours or days), and retrieved later as they are needed.

Looking at Data The Lisp Machine provides a high-resolution monitor that is useful for displaying data. A *layout* is a collection of *displays* of information that occupy an entire screen. Associated with each display are a number of *overlays*, which may be graphs, scales, or other attributes, such as a phonetic transcription overlaid on a spectrogram.

Spire allows users to compose their own layouts for specific research needs. Figure 1 shows an example of such a layout. The figure displays the wide-band spectrogram of the utterance, the zero-crossing rate, the original waveform, the orthographic and phonetic transcriptions, the narrow-band spectrum, and the LPC spectrum. This layout illustrates some of the interactive features of *Spire*. First, the user has direct control over all relevant display parameters. Thus, for example, the zero-crossing rate is displayed on the same time scale as the wide-band spectrogram. Second, all displays are time-synchronized. Moving a cursor in one display causes the other displays to change accordingly. Third, displays can be overlaid, and the display parameters of the overlay can be changed as well. For example, the LPC spectrum, overlaid on the narrow-band
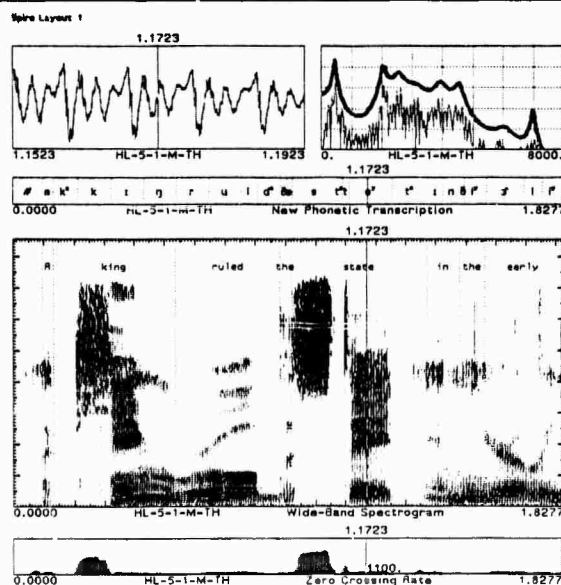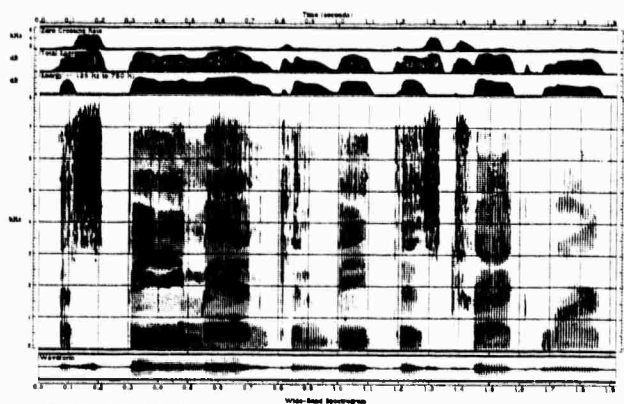

Figure 1: A *Spire* layout.

111

Figure 2: A high-quality spectrogram produced by *Spire*.

spectrum, is distinguished from the latter by a difference in line thickness.

Hard copies of displays can be obtained by a screen hardcopy command. In addition, a special higher-resolution hardcopy spectrogram format is available. Standard Lisp Machine hardcopy protocols are used, so the printer may be of any type. MIT has a Versatec V-80 printer/plotter with a resolution of 200 pixels per inch. Figure 2 is an example of a Versatec spectrogram.

Labeling Data *Spire* provides a convenient mechanism for users to enter an orthographic or phonetic transcription and time-align it with the speech waveform. The mouse is used to mark specific regions of the waveform and to associate each region with a label. An experienced acoustic phonetician can align a two-second utterance using *Spire* in about five minutes.

While manual time-alignment using *Spire* is quite efficient, it still requires the knowledge of a small group of experts. As a result, the amount of data that can be collected and aligned is greatly limited. In addition, phonetic alignment is often subjective, leading to inconsistencies among transcribers. The tedious nature of the task also tends to introduce human errors. We have recently extended *Spire*'s basic capabilities by developing a semi automatic system to perform the time alignment. The results of our preliminary evaluation are encouraging. For a description of the alignment system, see Leung and Zue [4] and Leung [3].

Search

In the search for relationships between the acoustic properties of speech and the underlying linguistic forms, speech researchers typically begin by examining a small number of utterances using tools such as *Spire*. Guided by their knowledge and intuition, they may propose an acoustic measurement that is potentially useful for identifying a particular phonetic event. Once such a hypothesis has been established, they must then apply the measure to a large body of data, either to validate assumptions derived from the limited data set, or simply to characterize

the performance of the features. At this point, a relatively sophisticated battery of statistical techniques is most useful.

Search (Structured Environment for Assimilating the Regularities in speeCH) is an interactive tool designed for exploratory analysis of speech data. *Search* consists of a set of statistical tools that operate on data generated by *Spire*. This software allows the user to gather statistics on thousands of tokens taken from hundreds of utterances. It has extensive graphics capabilities for displaying the data in various forms, including histograms, scatter-plots, and a bar-like display that allows users to view univariate distributions of data as a function of categorical variables (e.g., speaker sex or phonetic environment). *Search* also features a set of extensible data structures that form a convenient workbench for the design, implementation, and testing of various statistical algorithms.

The basis of *Search* is a set of algorithms for automatically designing classification trees from a learning sample. The primary method is the *CART* (Classification And Regression Trees) algorithm [1], a supervised classification algorithm that generates a binary decision tree using a maximum-entropy reduction criterion. An alternative method is an *ISODATA* cluster analysis routine that features a k-Means clustering algorithm. Both of these algorithms are optimization techniques that attempt to organize speech knowledge automatically. They reflect our emerging philosophy that researchers should approach the speech analysis problem with as much intuition as they can develop, and then allow the data and statistics to fill in any gaps in knowledge.

Search can also be used simply as a program for partitioning the data in terms of a set of criteria, and then displaying the data in a variety of ways. Figure 3 compares the distribution of duration for voiced and voiceless fricatives. The mean, the standard deviation, and the sample size for each class are indicated next to the bar-like displays. The data confirm the fact that voiced fricatives are generally shorter than voiceless ones, although there is substantial overlap due to the phonetic environments
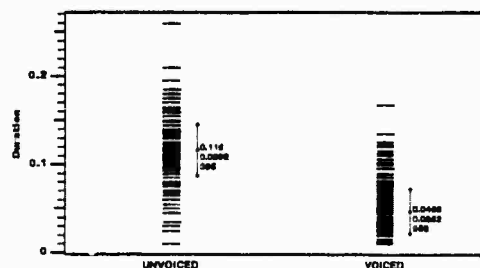

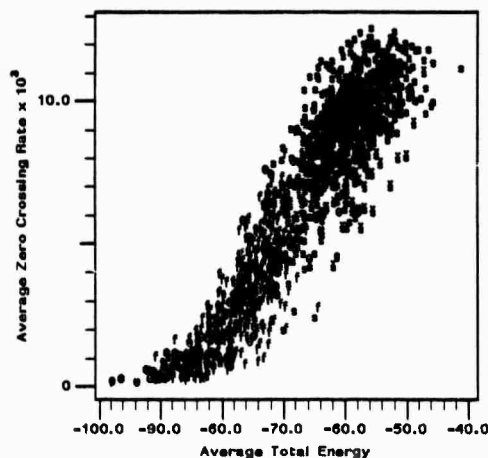**Figure 3:** Duration distribution produced by *Search*.

112

Figure 4: Zero-crossing rate/energy scatter plot produced by *Search*.

in which the fricatives appear. Figure 4 shows a scatter plot of zero-crossing rate versus total energy for the voiceless fricatives. We see that these two parameters are highly correlated; the strident fricatives /s/ and /š/ appear mostly in the upper right quadrant, and weak fricatives /f/ and /θ/ appear mostly in the lower left quadrant.

ALexiS

A language is limited not only by the inventory of basic sound units, but also by the frequency of usage and the allowable combinations of these sounds. With the availability of large and powerful computers, it is now possible to discover and quantify such distributional and sequential constraints using a large body of speech data.

ALexiS is an interactive system that provides many options for studying and displaying the constraints of a lexicon. *ALexiS* enables users to determine the frequency with which sound patterns occur, to study the phonotactic constraints imposed by the language, and to test the effectiveness of phonetic and phonological rules. In addition, users can define new operations and integrate them into the program.

ALexiS operates on a corpus consisting of a list of words or a set of sentences. Words in the lexicon are usually represented in terms of spelling, pronunciation (including syllable and stress markers), and other corpus-specific features such as the frequency count based on the Brown Corpus. User-specified constraints can be applied for each of these features, leading to a list of all words in the corpus matching the constraints.

Once the lexicon is specified, *ALexiS* can analyze the corpus in a number of ways. For example, users can generate a frequency distribution of words in the corpus in terms of the number of syllables, the stress patterns, or a particular sound pattern. As an example, Figure 5 shows
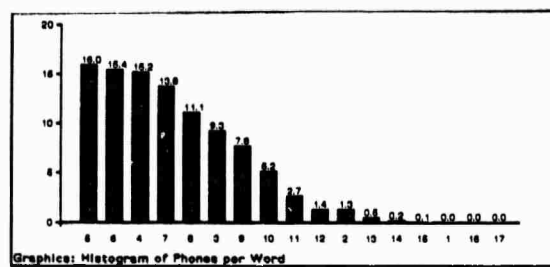

Figure 5: Histogram of word length in phonemes produced by *ALexiS*.

a histogram of the number of phonemes per word for the Merriam-Webster Pocket Dictionary.

Synth

Synth is the Klatt cascade/parallel speech synthesizer implemented as a *Spire* subsystem. Parameters that can be manipulated include fundamental frequency; formant frequencies, bandwidths, and amplitudes; amplitudes of voicing, frication, and aspiration; and frequencies and bandwidths of nasal and glottal poles and zeros.

The computations are done mainly in the FPS, and therefore the synthesis process is reasonably fast (approximately 15 times real time). The user interface is a special *Spire* layout, as shown in Figure 6. The user specifies time-value pairs for the parameter tracks, either with the mouse or from text. For example, to enter a track for F_2, the user selects F_2 from the menu on the left, and a display of the three formant tracks appears. The user can then enter a new track for F_2 or modify the existing track. Default parameter track layouts satisfy most users' requirements; however, users are also free to design their own special-purpose layouts. Once a set of tracks has been completed, the user selects "Synthesize" on the synthesizer menu, and a speech waveform is generated from the track data. This waveform is now available on the *Spire* utterance list and can be treated like any other *Spire* utterance. Thus, for example, a wide-band spectrogram and LPC spectrum of the synthetic utterance can be displayed, along with the same attributes for a similar natural utterance.

STATUS REPORT

Spire

Spire has been carefully evaluated and refined over the last several months. A number of improvements have been made to the system, including the following:

- It is now possible to save computed parameters separately from permanent parameters such as the waveform. This reduces the chance that "permanent data" will be accidentally destroyed (as has happened) and also reduces name conflicts among different users.

- Each Unibus interface now has better support. The FEP interface code has been modified to make it more stable in shared Unibus environments (such as a Lisp Machine or a Vax). The G-Bus interface
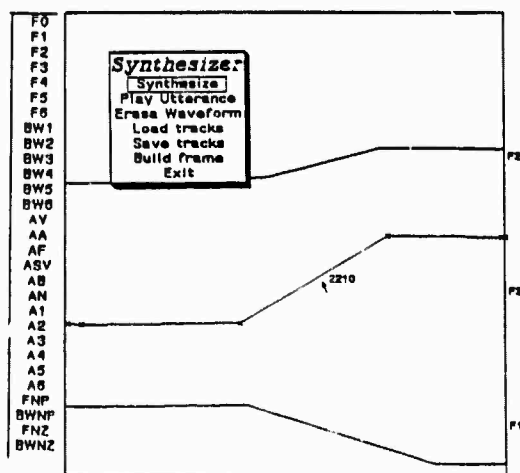
Figure 6: Typical Synthesizer Layout

seems to transfer data twice as quickly as the FEP interface.

- Utterances can now be played through the console when the Lisp Machine has audio capabilities, although the method used does not provide accurate reproduction.

- Lisp-based FFT and LPC are now available. With the availability of these programs, it is now possible to run *Spire* on a stand-alone Lisp machine with no array processor.

- A simple waveform editor has been added.

- The hand-alignment of transcriptions has been greatly simplified.

- The user interface is more consistent, making it easier for users to extend *Spire* by adding commands. The same command processor used by the Lisp Machine is now used by *Spire*.

Another task we have completed is evaluation of the FPA. Although for straight signal processing the FPS is still much faster than any other alternative, an FPA does cut times roughly in half. Table 1 gives some illustration of the relative timing for a Lisp Machine with different configurations. The energy computation is based on the computation of the energy in the range from 0 to 5,000 Hz, computed once every 5 ms. from speech sampled at 16,000 Hz. The spectrogram was computed from the speech waveform with a 383-Hz analysis rate. The numbers in Table 1 represent processing time (in seconds) per second of speech.

Finally, two documentation efforts for *Spire* are under way. A user's guide to *Spire*, aimed at the beginning user, has been written and is currently being reviewed. The final version should be available within the next month.

Table 1: Timing Comparison for Lisp Machine with Different Configurations

Configuration	Energy	Spectrogram
LM alone	27.6	63.0
LM + IFU	24.0	53.1
LM + FPA	11.8	31.5
LM + FPA + IFU	8.3	21.8
LM + FPS	1.6	3.5

A reference manual has also been prepared; a copy of this document, along with the source codes, is to be distributed with the current release of *Spire*.

Search

Search has undergone extensive redevelopment in recent months. Although the exterior aspects of the program (i.e., the graphical displays and the user interface) have remained more or less unchanged, we have redesigned a major portion of the system internals, including a complete overhaul of basic data structures. In instituting these changes, we have made every effort to keep major differences from affecting the user. For example, our basic data structure is the sample, which contains a collection of tokens. Although the sample has been reimplemented, most previous code written to extract data from samples and to manipulate samples should still work.

At present, the system is considered experimental and still somewhat fragile. We are now making *Search* available to other users in our own laboratory, so that we may get feedback and correct problems before beginning wider distribution. We expect that in two to three months, *Search* will be ready for release to other sites.

We are, however, only now starting to create documentation for the software. As this is expected to be a somewhat lengthy process, it is likely that early releases of *Search* will not have accompanying documentation.

ALexiS

Although *ALexiS* is still in the developmental stage, it has been used for a number of tasks. In addition to use with Japanese lexicons and in teaching, it provided the framework for the design and analysis aids used in the acoustic-phonetic database project. These applications helped test the design of *ALexiS* for extensibility and uncovered a number of weaknesses.

In the near future these design flaws will be corrected and the implementation will continue. By using the program in its uncompleted state, we hope to provide a better, more flexible system while avoiding untested design hypotheses.

Synth

The *Synth* system is nearing its final stage of development. It is currently being evaluated by a number of users at our laboratory and should be ready for release by mid-

to late spring. Documentation and a user's guide are being developed concurrently, but may not be available at the time of the initial release. However, user interaction with the system is essentially the same as user interaction with *Spire* and should require little new instruction.

While *Synth* is limited at present to systems whose hardware includes an FPS, future revisions should remove this constraint. A further change in *Synth* that is now under discussion is improvement of the system's modularity, i.e., allowing user-designed modules to replace those provided.

SUMMARY

This paper describes a set of research tools being developed in the Speech Group at MIT. Together these systems provide a unified environment that enables speech scientists to move from one task to another. For example, users can explore the statistical properties of a large body of data using *Search*, and then directly enter *Spire* to examine specific outlier tokens. An integral part of the system is the maintenance of a large database of more than three hours of digitized speech. Most of the utterances have been transcribed, and the transcriptions have been time-aligned with the corresponding waveform.

The development of these research tools is an ongoing process. Our goal is to create a research environment that is easy to use, thereby increasing the amount of data that speech scientists can examine and, as a consequence, extending our knowledge about speech. The Lisp Machine workstation and related software systems are playing an important role in advancing our understanding of the acoustic properties of speech sounds.

While these systems are still being actively improved, and not all the software is widely available, members of the Speech Subsystem of the DARPA Strategic Computing Program may contact us directly to obtain further information on acquiring the software.

ACKNOWLEDGMENT

We gratefully acknowledge the contribution of David W. Shipman, who designed and implemented the original *Spire* and provided many helpful ideas for developing the software tools.

REFERENCES

[1]   Breiman. L., J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Belmont, CA: Wadsworth International Group, 1984.

[2]   Cyphers, D. Scott, "Spire: A Speech Research Tool," S.M. thesis, Massachusetts Institute of Technology, May 1985.

[3]   Leung, Hong C., "A Procedure for Automatic Alignment of Phonetic Transcriptions with Continuous Speech," S.M. thesis, Massachusetts Institute of Technology, January 1985

[4]   Leung, H. C., and V. W. Zue, "A Procedure for Automatic Alignment of Phonetic Transcription with Continuous Speech," *Proc. ICASSP 84: IEEE International Conference on Acoustics, Speech, and Signal Processing,* 1984, pp. 2.7.1-2.7.4.

OPTIMAL AND SUBOPTIMALTRAINING STRATEGIES FOR AUTOMATIC SPEECH RECOGNITION IN NOISE, AND THE EFFECTS OF ADAPTATION ON PERFORMANCE

Janet M. Baker and David F. Pinto
Dragon Systems, Inc.
Chapel Bridge Park
55 Chapel Street
Newton, Massachusetts, 02158, U.S.A.

ABSTRACT

The quality of operational speech recognition performance in the presence of variable ambient noise can be significantly affected by the conditions under which patterns are initially trained, as well as by subsequent modifications of these patterns through "adaptation". These effects are clearly evidenced in a series of experiments conducted using the Dragon Systems Speech Driver in conjunction with DragonLAB (an experimenter's workstation facility), commercially available with the IBM PC Voice Communications Option and Voice Recognition Tool Kit, running on an IBM PC/AT. For each of two discrete command/control vocabularies, "Menu" (24 words) and "DOS" (29 words), training and multiple test session recordings were made at each of 55 and 65 dB ambient noise levels with an inexpensive cassette tape-recorder microphone, and at 85 dB with a close-talking noise-cancelling microphone.

The seven American English-speaking subjects (4 male, 3 female) included in this database, exhibit diverse voice qualities, dialects, speech recognition familiarity, etc. A total of over 20,000 training and test utterances were collected for this database.

The results of this series of experiments demonstrates the effects on recognition performance of different training sets on the same test sets, and the effects of adaptation (in a supervised learning mode), both for speaker-dependent and cross-speaker modes. For all speakers and both vocabularies, the best speaker-dependent recognition is consistently obtained when training and test materials are recorded at the same noise level.

Much of the deterioration of performance across different training and test conditions can be reduced by employing supervised adaptation during the course of the recognition tests themselves, with each new test token subsequently being used as additional training to adapt its model.

Empirical data demonstrates the high variability of performance on cross-speaker recognition where a given speaker's test sets are recognized using a different speaker's training data. More interesting, though, is that the cross-speaker performance obtained by using on-line supervised adaptation during recognition tests, can readily approach or equal pure speaker-dependent performance, after adaptation on only a very few tokens. Furthermore,

preliminary evidence indicates that speaker-dependent patterns generated for a given speaker, and then adapted with a comparable amount of speech training data from another speaker, may continue to be used to achieve good performance on the initial speaker.

In short, the experiments presented here demonstrate the effects of alternative training protocols on recognition performance under operationally-oriented conditions. The effects on performance of even severe deleterious training-test mismatches due to noise, microphones, and speakers may be substantially reduced by applying supervised automatic adaptation techniques during recognition. This paper further shows that using even quite limited amounts of speaker-specific information for adapting pre-existing patterns derived from other speakers, can provide striking performance improvements. These experiments indicate that speaker-independent or cross-speaker systems without adaptation, are likely to perform much more poorly than similar systems capable of capturing new speech data and automatically adapting to new speakers as they use the system. As with other endeavors, learning from experience improves future performance expectations!

BACKGROUND

There are two major types of speech recognition tests: 1) benchmark tests (often with very quiet sound booth recordings), and 2) operational or applications-oriented tests designed to understand and predict performance under actual field use. Although these may in part overlap, both have distinctive roles and values associated with them. Benchmark tests typically assess a well-circumscribed capability that is acknowledged to indicate certain basic technology strengths and weaknesses. Furthermore, these tests often derive additional value because they lend themselves to appropriate comparisons of alternative technical approaches and implementations.

The 20-word discrete utterance Texas Instruments database [1] is an example of just such a benchmark test. Benchmark tests also may push system limits along certain dimensions somewhat further than may be encountered under normal operational use. For speech recognizers, this may require the discrimination of highly confusable or minimally distinctive acoustic cues, without the contextual or other information usually available in a typical

application.

Applications-oriented databases, on the other hand, test specific conditions anticipated or encountered operationally. For this study, two different scripts, named "Menu" and "DOS", were selected as representative of different typical end-user applications for voice command/control of popular software. Three databases were constructed for each of the two scripts with training and test utterances obtained under moderately quiet to much more severely noisy conditions. Speakers were instructed to speak utterances as soon as they were prompted on a monitor. Although the 7 speakers chosen to participate in these databases varied significantly with regard to their previous experience with recognizers, all participants were judged cooperative. The effects of additional variables such as cognitive loading, explicit psychological or physiological stress, speech pathologies, language difficulties, etc. were not assessed by these tests. For all six of these databases, training and test data were recorded at different sessions; multiple test sessions at any given noise level were recorded on different days.

Both the Menu and DOS training and test scripts were recorded for all speakers in moderate ambient noise (55dB and 65dB) with an inexpensive cassette tape-recorder hand-held microphone, as well as in much more severe noise (85dB) with a high quality noise-cancelling headset microphone. The noise itself was generated by exposing the speakers to specified levels of competing background trade show and other noise. At audible levels, such competing speech can be far more problematic and challenging to automatic speech recognizers, than "white", "pink", or other steady state noise types at higher levels.

Competing speech at progressively higher levels has the potential for progressively greater interference, especially as the signal-to-noise ratios degrade. Human speakers trying to communicate verbally under noisy conditions, routinely modify their speech to help ensure its intelligibility. Speaking louder, more slowly, and articulating more carefully, are well-known compensatory behaviors. For superior recognition performance, the speech data obtained during training should be similar to that anticipated under normal use [2,3]. An appreciation of this phenomenology explains why gross mismatches of training and test conditions, or "additive-noise test simulations" (using "quiet" speech training for recognizing test data artificially mixed with additive noise) in the absence of adaptive training mechanisms, almost invariably lead to suboptimal recognition performance bearing little relevance to "state-of-the-art" applications. The negative effects of severe training-test mismatches (across very different noise levels, microphones, and speakers) are shown by this series of experiments, as are the substantial improvements that can be realized through on-line adaptation even under such undesirable conditions.

RECORDING/RECOGNITION EQUIPMENT

For the Menu and DOS databases, live input to a Sony Digital Audio Processor PCM-F1, was entered through a Radio Shack microphone #33-10400 at levels of 55 and 65dB, and a Shure SM-12A microphone at 85dB. An Apple IIE computer was used to prompt the speakers and control a Panasonic NV8200 Video Cassette Recorder, for recording the speech data on Maxwell Epitaxial HGX video tape. Testing was also performed on the Texas Instruments isolated word database by converting analog tapes to digital format by playing the tapes on a Tandberg Two-track Series 15 reel-to-reel tape recorder to the Sony PCM-F1. The recognition tests were performed by playing the video cassettes on the Panasonic VCR through the Sony PCM-F1 to an IBM PC/AT with the IBM PC Voice Communications Option board and software installed.

The digital processor and board were connected through an attenuator. Attenuation settings were adjusted once separately for the T.I. database and the Radio Shack and Shure microphones, so that the amplitude coming in to the IBM Voice Communications Option card would be comparable to the amplitude produced by a live speaker.

Once sessions were recorded on tape, the IBM Voice Tool Kit was used to digitize these results at 8kHz onto the AT's hard disk, so that recognition tests could be run quickly, accurately, and reproducibly . Digitizations were also stored on high density floppy disks.

BACKGROUND NOISE

Background noise was generated with an audio tape of trade show speech and noise played on a Nakamichi BX-1 cassette recorder over a Fostex 6301 Personal Monitor. Standard volume control settings on the monitor, adjusted the level of the noise. The noise itself consisted largely of background conversations from a trade show in a large convention center. In addition to steady on-going intelligible conversations around the recorder, there are sporadic broadcast announcements, bells, and impulse noise from the public address system. Occasionally passersby shouted into the microphone. This causes the actual noise level to vary above the noise level required for the test. A Bruel & Kjaer Precision Integrating Sound Level Meter Type 2221 was used for the sound level measurements. The meter was placed at the same height as the speaker's mouth and positioned at the same distance from the monitor as the speaker's mouth in an equilateral triangle configuration. Noise levels were determined by repetitively measuring the peaks of successive 10 second intervals over several minutes.

SCRIPTS

The Menu and DOS scripts represent two vocabularies typical of those used for actual command/control and data entry applications. The DOS vocabulary contains utterances of varying lengths: e.g. "tree" and "make_directory". Some explicitly confusable word pairs and multiples have been deliberately included, such as 1)"directory", "make_directory", "change_directory", and "remove_directory", and 2)"copy", "compare", "disk_copy", "disk_compare", "check_disk", etc. In all of these, significant

portions of competing function names are identical, thereby increasing their potential for acoustic confusability with each other. The potential for recognition errors is enhanced in the presence of noise, especially speech noise, which can best mask the remaining acoustically distinctive portions. On the other hand, to comply with high performance application design principles, the highly confusable rhyming letter names of the natural alphabet were replaced by the corresponding international communications alphabet designations ("alpha","bravo",...,"zulu") in the Menu script.

The DOS script consisted of enter, cancel, menu, DragonKEY, alpha, bravo, charlie copy, date, time, delete, directory, erase, print, make_directory, change_directory, remove_directory, rename, tree, type_file, disk_copy, backup, check_disk, compare, disk_compare, format, restore, and volume. Menu's script contained the ten digits zero through nine, as well as alpha, bravo, charlie, delta, echo, foxtrot, golf, hotel, india, juliet, zulu, DragonKEY, escape, and enter.

For training at 55 and 65dB, six tokens were collected in a sequential manner (a1,b1,...,n1,a2,...,n5,a6,...,n6). For the 85dB noise level, ten training tokens were collected to provide better sampling of the noisy speech data. The test scripts recorded at all noise levels, contained three utterances of each word, also recorded sequentially. The training sets contained a total of 7102 tokens, and the test sets contained a total of 12955 tokens.

The TI script contains 10 command/control words (yes, no, erase, rubout, repeat, go, enter, help, stop, start) and the 10 digits zero through nine, spoken in a sound booth, by 16 speakers (8 males, 8 females) during 8 recording sessions. For each of the 20 words, 10 training tokens are provided (3200 training tokens); the test set consists of 16 tokens for each of the 20 words for a total of 5120 test tokens. Although the recording environment was exceptionally quiet, some of the speakers deliberately introduced significant variations in pronunciations and prosodics. In the preparation of this data, splices of digital silence were interspersed between the speech utterances creating occasional soft clicks.

RECORDING PROCEDURE

Subjects sat in a typical, acoustically untreated office. 2 feet from the noise source, and watched an Apple 11E monitor for prompts. The noise speaker itself faced the wall at a distance of 7.5 inches to reflect the noise from multiple room surfaces. Prompts were spaced at 2 second intervals. After each sixth prompt, the system paused to allow the user to re-record the last six utterances (to correct speaking errors) or to continue. Each session typically consisted of six recordings, for the 2 scripts at each of the 3 noise levels. At the 85dB level, most subjects chose to wear simple moldable wax ear plugs. Although discomfort due to the high noise was somewhat alleviated thereby, the perceived noise level, even with ear plugs, was significantly higher than at 65dB, without ear plugs. All subjects participated in from three to

six separate recording sessions, varying between morning and afternoon sessions on different days. Two of the subjects BP (male) and HM (female) did not participate in the 85 dB recording sessions.

SUBJECTS

Participants included 4 males and 3 females ranging in age from 25 to 42 years, with dialectical characteristics ranging from east coast to midwest. These individuals were specifically chosen for their diversity of speaking styles, rates, voice qualities, and their prior experience with speech recognizers (2 none, 2 some, 3 extensive).

BASE PERFORMANCE RESULTS

Dragon Systems discrete recognition implemented on the IBM Voice Communications Option board and software, has been tested on 3 scripts, under widely variant noise conditions. These databases were selected and designed to provide complementary information on multiple aspects of recognition relevant to a range of benchmark and operationally oriented conditions. The following table summarizes these results.

SUMMARY TABLE		
TABLE I ORAGON RECOGNITION RESULTS (% CORRECT)		
MEAN SPEAKER T.I. 99.5		
MEDIAN SPEAKER T.I. 100.0		
* * *	* *	*
MEAN SPEAKER	55DB	65DB	85DB
DOS	99.4	99.8	99.4
MENU	99.1	99.1	99.2
* * *	*	*	*
MEDIAN SPEAKER		
DOS	99.8	99.8	100.0
MENU	99.5	99.7	100.0

To obtain these results, a total of 10,302 training utterances and 18,075 test utterances were processed. Both median and mean performance results are provided here for comparison. From a statistical point of view, the results of all these tests are so close to 100%, that statistically significant differences between the individual tests cannot be reliably observed [5]. Several interesting data tendencies bear discussion, however.

On the T.1. 5120 token test database, 25 errors were observed; 19 of the 25 errors were confusions between "no" and "go". On 8 of the speakers, no errors were observed. The observed errors were divided equally between 4 men (13 errors) and 4 women (12 errors).

Test results on the 29-word DOS script were consistently higher over the 55dB, 65dB, and 85dB ambient noise levels than the results obtained on the 24-word Menu script. Although the DOS script has a higher branching factor, it has only 4 monosyllabic words as compared to 8 in the Menu script, and may be easier to recognize.

One also notes that for both of these scripts, the observed performance actually improves at the higher background noise levels! An appreciation of human communicatory behavior may explain this

counterintuitive result. When people communicate verbally in noisy environments, they can improve the intelligibility of their speech by speaking more loudly and slowly, as well as articulating more carefully. Given an adequate signal-to-noise ratio, the utterances spoken by people employing such compensatory behaviors, may well contain more acoustic information, and therefore be easier to recognize for both human listeners as well as speech recognizers, as compared to more casual speech spoken in quieter environments.

Obviously we are not implying generally that there is a positive correlation between performance, and increased vocabulary size or ambient noise levels... We note however that under the range of conditions tested by these databases, minor differences in vocabulary size or major differences in noise levels are not the limiting factors for these algorithms. Considerations of acoustic confusability of competing utterances and the human factors of speaking behaviors appear to exert a greater effect on performance here, and ought to be taken into account in the design of operational applications. Analysis of the experimental results for these speakers individually, reveal no obvious performance correlations with gender, age, voice quality, dialect, speaking style/rate, previous speech recognition experience, A.M. vs. P.M. recordings, or the number of elapsed days between training and test sessions.

SUBOPTIMAL TRAINING

The results reported above, indicate recognition results when both the training data and the test data are collected under similar conditions. The question is frequently raised re: the performance cost of using recognition under conditions not actually represented in the training data. The DOS and Menu databases collected here, reflect separately, in their training and test sets, 3 distinctive noise levels and 2 microphones with very different design characteristics. These provide an opportunity for addressing such issues. The second question is, despite severe deleterious effects resulting from inadequate/inappropriate training, what can be done to improve recognition performance.

Response to Question 1: The following tables list the mean percentage of errors observed speaker-dependently, with each training set (at 55, 65, and 85 dB noise levels) tested against each test set (at 55, 65, and 85 dB noise levels). Each table entry represents the results on all the collected data, with
set sizes ranging from 1608 to 2552 test tokens.

TABLE II						
Mean Recognition Error Percentages						
	Menu (24 words)			DOS (29 words)		
Test Noise:	55dB	65dB	85dB	55dB	65dB	85dB
Training Noise:						
55dB	.9	1.6	30.3	.6	.6	20.0
65dB	1.7	.9	27.6	1.2	.2	27.0
85dB	16.7	22.6	.8	15.0	17.8	.6

The most severe degradations occur when the microphones used in recording the training and test sets are different designs (i.e. one is the Radio Shack omnidirectional hand-held mike, and the other

is a Shure noise-cancelling headset mike). The worst results of all are obtained with training at 55 dB (Radio Shack mike) and testing at 85 dB (Shure mike).

Response to Question 2: The standard Dragon training algorithms not only provide a mechanism for building stochastic models from a specified number of training tokens, but also allow for on-line adaptation to further refine these models as more speech samples become available [6]. The contributions of these incremental samples are weighted according to the total number of speech samples represented by a given model. "Supervised" adaptation allows new speech utterances to be adapted into existing models. In operational applications, supervised adaptation implies that the identity or label for each utterance has been confirmed. Unsupervised adaptation would permit the adaptation of models corresponding to the recognition result or label of a test utterance, regardless of its correctness.

The results of running automatic supervised adaptation on-line, during recognition, are very effective in reducing the percentage of errors observed under the 55 dB training/ 85 dB test paradigm described above. The amount of improvement obviously can vary according to a number of criteria. The number of additional speech tokens adapted in, appears to steadily improve performance for the 5 speakers represented here. Their individual test sets provided a maximum of 6 to 20 new speech tokens for adaptation. However, even with extensive adaptation, resulting in major reductions in the percentage of errors observed, the degradation in performance is quite marked, relative to properly matched training and test conditions with the same microphone type and noise level.

TABLE III
55dB Training/85dB Test, With and Without Adaptation
MEAN RECOGNITION ERROR PERCENTAGES

	DOS				MENU			
	# adapt. tokens	with adapt.	w.o. adapt.	tr 85dB/ test 85dB	# adapt. tokens	with adapt.	w.o. adapt.	tr 85dB/ test 85dB
OP	6	39.7	49.4	0.0	6	48.6	54.9	0.0
IJ	9	8.0	13.8	0.0	9	23.1	41.7	0.0
GL	16	4.3	16.4	2.6	16	9.1	17.7	3.4
AJ	16	6.5	18.7	0.0	19	26.6	35.2	0.0
PD	20	.2	16.4	0.0	20	5.2	22.7	0.0

CROSS-SPEAKER TESTING

Another means of mismatching training and test conditions, is to recognize one speaker with models derived from another. As such, this is speaker-independent recognition, but there is no presumption that the initial models used for recognition, adequately represent any given, let alone broad, population of speakers.

In this set of experiments, recognition tests will be run first without, and then with supervised adaptation. All training and test sets are those collected under the 55 dB conditions. It is well known that performance results on a speaker using models from (an)other speaker(s) are likely to be highly variable. This is easily demonstrated. Less well appreciated is the non-commutability in performance of recognizing speaker A with speaker B's models, as compared to recognizing speaker B with speaker A's models. Examples are furnished in the following table (Menu database) for male

speaker GL tested with another male speaker BP, and then tested with a female speaker HM. Controls are provided by testing each of the speakers with their own models, as noted.

TABLE IV

55dB MEAN RECOGNITION ERROR PERCENTAGES

MENU (No adaptation)	CROSS-SPEAKER		SPEAKER-DEPENDENT	
TEST SPEAKER	ENROLLED SPEAKER	ERROR PERCENTAGE	ENROLLED SPEAKER	ERROR PERCENTAGE
GL	BP	4.7%	GL	.3
BP	GL	17.4	BP	1.4
HM	GL	33.3	HM	.5
GL	HM	15.8		

ADAPTING TO A NEW SPEAKER - QUICKLY

With supervised adaptation, as previously described, each test token is first subjected to testing, and then it is used as a training token for incremental model updating. All test tokens are presented once and only once, whether or not they are subsequently used for model refinement. In running these experiments, it quickly became apparent that even a small amount of adaptation rapidly improved recognition. Marked reductions occurred in the number of observed errors despite the conservative weight accorded to additional incremental training tokens. Recall that the original speaker models were generated from 6 tokens, and that consequently the first new token from a different speaker would only be acr ed a weight of one seventh, the second token right of one eighth, etc.

The following table records for both the Menu and DOS databases, the number of observed errors after each new "run". A run contains 1 instance of each vocabulary word, with a run containing a total of 24 utterances for the Menu vocabulary, and 29 utterances for DOS. Note that for all of the nine cross-speaker tests noted here, recognition performance with adaptation appears effectively to have converged to typical speaker-dependent performance after 5 or fewer new tokens. Furthermore after only 1 new token is adapted in, the error rate (compared to that obtained without adaptation) is halved; after the 2nd token is adapted in, the error rate is halved again!

TABLE V

ERRORS RECORDED ON A RUN-BY-RUN BASIS FOR
CROSS-SPEAKER RECOGNITION WITH ADAPTATION

ENROLLED SPEAKER	ADAPTED SPEAKER	# TOKENS ADAPTED TO NEW SPEAKER										
		0	1	2	3	4	5	6	7	8	9	10
MENU												
AJ	BP	3	3	1	-	1	1	-	-	-	-	-
BP	GL	9	2	2	2	-	-	-	-	-	-	-
AJ	HM	4	3	-	-	1	1	-	-	-	-	-
GL	BP	-	-	-	-	-	-	-	-	-	-	-
GL	HM	4	2	1	1	-	-	-	-	-	-	-
DOS												
GL	HM	4	6	3	4	3	1	1	-	1	-	-
GL	BP	-	-	-	-	-	-	-	-	-	-	-
GL	HM	4	2	-	-	-	-	-	-	-	-	-
AJ	BP	5	-	-	-	-	-	-	-	-	-	-
Total # errors		33	18	7	7	5	3	1	-	1	-	-

An abridged version of this paper will appear as titled, in the IEEE Proc. of ICASSP-86, to be presented in Tokyo, Japan, April 1986.

Another question to ask is what happens to recognition performance on a given speaker after their models have been adapted in varying degrees, to another speaker. Some data is provided indicating degradation in performance, gradually increasing as the total number of adapted foreign speaker tokens appreciably supersedes the number of speaker-dependent tokens. It is worth noting however, that as seen in the immediately preceding experiments, these adapted models also provide for speaker-dependent-like performance for that other speaker. As previously shown, cross-speaker recognition without adaptation, is far worse, of course. It is our view that understanding the limits on adaptation, and performance trade-offs in extending multi-speaker models to larger speaker populations, should form the basis for future research investigations.

TABLE VI

ERROR PERCENTAGE OBTAINED BY
RERUNNING INITIAL SPEAKER AFTER
N RUNS OF ADAPTATION TO SPEAKER GL

INITIAL SPEAKER RERUN	# RUNS ADAPTED TO GL			SPEAKER - DEPENDENT CONTROL W.O. ADAPT.
	5	10	15	
HM	.5	.7	.9	.5
BP	2.1	2.1	4.1	1.4

CONCLUSIONS

These experiments obviously have a number of inherent limitations re: number/choice of speakers, specific recording conditions, etc. Nonetheless the results obtained here strongly suggest that under training/test mismatch situations, especially across different speakers, the recognition performance obtained in the absence of supervised adaptation is far inferior to that obtained with even a small amount of adaptation. Alternatively, significantly higher recognition performance can be achieved by automatically integrating new information as it becomes available during actual system use. Adaptation thus provides an operationally effective bridge spanning the extremes of speaker-dependent and speaker-independent recognition. This productive continuum richly deserves to be better explored!

BIBLIOGRAPHY

[1] Doddington, G.R. and Schalk, T.B., "Speech recognition: turning theory to practice", IEEE Spectrum 18(9), Sept., 1981.
[2] Rollins, A. and Wiesen. J., "Speech recognition and noise", Proc. of IEEE ICASSP-83, Boston. 1983.
[3] Kersteen, Z.A.. "An evaluation of automatic speech recognition under three ambient noise levels", Proc. of NBS Workshop on Standardization for Speech I/O Technology, March. 1982.
[4] Pallett, D.S., "Performance assessment of automatic speech recognizers," J.Res.NBS, Vol. 90, No. 5, Sept-Oct., 1985.
[5] Baker, J.M., "The performing arts--how to measure up", Proc. of NBS Workshop on Standardization for Speech I/O Technology, March. 1982.
[6] IBM PC Voice Communications Application Program Interface (API) Reference, 1985.